UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE ESTATÍSTICA E INVESTIGAÇÃO OPERACIONAL

A Heuristic Approach for a Multi-Period Capacitated Single-Allocation Hub Location Problem

Ana Margarida Cordeiro de Sousa Wemans

Mestrado em Estatística e Investigação Operacional
Especialização em Investigação Operacional

Dissertação orientada por:
Professor Francisco Saldanha da Gama

2016

ACKNOWLEDGMENTS

First of all, I would like to thank God for the opportunity and the capacities to do this dissertation.

I would like to express my gratitude to my dissertation advisor, Professor Francisco Saldanha da Gama, for his collaboration and for all I learned from him.

I would also like to thank all the Professors that accompanied my academic path for all the knowledge transmitted.

A big thank you to my colleagues and friends for their ideas, support and interest, especially to Catarina Mateus and Lukáš Zibala.

Finally, I must express my gratitude to my family, for all the patience and help while I was doing my thesis.

Thank you very much.

Ana Wemans

RESUMO

Hubs são instalações centrais que funcionam como pontos de consolidação de fluxo, ou, como definido por Alumur e Kara [3], instalações especiais que servem como pontos de troca, *transshipment* e triagem em sistemas de distribuição de muitos para muitos. Muitas vezes são usados para tirar partido de factores de desconto e economias de escala associados à consolidação de fluxo. Normalmente, é mais eficiente consolidar o fluxo proveniente de localidades próximas em vez de ligar directamente cada par Origem-Destino (O-D). O uso de redes de hubs é bastante relevante em sistemas logísticos de distribuição, redes de transportes públicos e serviços de correio, por exemplo.

Dependendo da natureza do problema em estudo os hubs podem realizar diferentes funções, como troca, triagem ou ligação, permitindo que os fluxos sejam redireccionados, consolidação ou separação de fluxos, processamento do fluxo ou ainda divisão ou combinação de fluxos, como no caso das redes de telecomunicações.

Um problema de localização de hubs consiste na escolha dos nodos onde será realizada a localização dos hubs e na alocação de nodos de procura a esses mesmos hubs de modo a encaminhar o fluxo entre os pares origem-destino. Na maioria dos casos o objectivo é minimizar o custo total envolvido.

Como Campbell e O'Kelly [17] realçam, algumas características distinguem os problemas de localização de hubs de outros problemas de localização. Num problema de localização de hubs (HLP) a procura está associada a fluxos entre pares O-D, os fluxos podem passar através dos hubs, a localização dos hubs tem que ser determinada, existe algum benefício ou obrigatoriedade em rotear os fluxos pelos hubs e o valor da função objectivo depende da localização dos hubs e do roteamento dos fluxos. Em geral, num problema de hubs fluxos directos entre pares O-D não são permitidos.

As principais decisões relacionadas com problemas de localização de hubs estão relacionadas com a localização dos hubs e o roteamento dos fluxos, incluindo as ligações entre hubs e os restantes nodos e as ligações entre cada par de hubs.

De modo a melhor interpretar a realidade, diversos tipos de problemas podem ser considerados, dependendo das suas características.

A rede constituída pelos hubs pode ser completa ou incompleta. Numa rede completa, todos os pares de hubs estão directamente ligados entre si. Numa rede incompleta, as ligações entre os hubs fazem parte do processo de decisão.

Num problema de localização de hubs diversas estratégias de afectação entre os nodos e os hubs podem ser consideradas, sendo as mais comuns *Single-Allocation* e *Multiple-Allocation*. No primeiro caso, cada nodo (não hub) deve estar afecto a exactamente um hub e no segundo, cada

nodo pode estar afecto a mais que um hub.

A maior parte da literatura relativa a HLPs considera problemas estáticos, ou seja, um plano de acção deve ser feito e implementado num único passo. Recentemente, algum trabalho tem vindo a ser desenvolvido sobre HLPs multi-periódicos. Neste caso, um horizonte temporal é considerado de modo a reflectir o tempo para implementar completamente a rede. Tipicamente, este horizonte temporal é dividido em diversos períodos de tempo. O objectivo é definir um plano multi-periódico para a localização dos hubs e o roteamento dos fluxos.

Nesta dissertação, o problema em estudo é um *Problema de Localização de Hubs Multi-Periódico com Capacidades Modulares*[1]. Na vertente estudada deste problema, considera-se que cada nodo deve ser afecto a exactamente um hub (*Sinlge-Allocation*), que existe apenas um tipo de fluxo (*Single-Product*), que a procura é determinística e que a rede a nível dos hubs é incompleta. Para além disso, consideram-se custos fixos e variáveis para os hubs, custos operacionais para as ligações entre hubs, custos fixos para a instalação de módulos de capacidades nos hubs e custos de roteamento de fluxos. O problema consiste em determinar quando e onde instalar hubs, determinar as afectações entre nodos e hubs em cada período de tempo, determinar as capacidades modulares a instalar em cada hub e período, determinar as ligações entre hubs usadas em cada período e determinar o roteamento dos fluxos na rede.

Em 2016, Alumur et al. [6] apresentaram uma formulação em programação linear inteira mista para este problema e, através da realização de testes computacionais, concluíram ser necessário desenvolver uma heurística ou algoritmo para encontrar soluções admissíveis de boa qualidade para instâncias de grande dimensão. O objectivo desta dissertação passa, precisamente, por desenvolver uma heurística para obter (boas) soluções admissíveis para este problema num espaço de tempo razoável.

Uma vez que, em problemas de localização, soluções estruturalmente diferentes podem ter custos muito próximos e vice-versa, a aplicação de um processo baseado em Pesquisa Local poderia gerar algumas dificuldades a nível da passagem de uma solução admissível para outra melhor. Para além disso, por causa das restrições de capacidade e de *Single-Allocation*, a utilização deste tipo de procedimentos poderia causar problemas ao nível da admissibilidade das soluções. De modo a evitar estas situações, optou-se pela aplicação de um algoritmo de *Kernel Search*.

Kernel Search é uma heurística baseada na ideia de identificar subconjuntos de variáveis e resolver uma sequência de problemas de programação linear inteira mista (MILP) restritos a esses subconjuntos de variáveis (Guastaroba e Speranza [34]).

As variáveis são divididas entre um *Kernel* e uma série de *Buckets* (ou "baldes"), por ordem de probabilidade de tomarem valores positivos na solução óptima. Note-se que esta probabilidade é apenas empírica. Considera-se que uma variável tem uma maior probabilidade que outra de tomar valores positivos na solução óptima se tiver maior valor na relaxação linear do problema. No caso de terem o mesmo valor, considera-se que é a que apresenta um menor custo reduzido que tem maior possibilidade de tomar valores positivos na solução óptima.

Este esquema heurístico é composto por duas fases: a fase de inicialização e a fase de solução. Na fase de inicialização, o *Kernel* e os *Buckets* são construídos, com base nos valores da relaxação linear do problema e um primeiro problema MILP restrito às variáveis do *Kernel* é resolvido. Na fase de solução, é resolvido um problema MILP restrito às variáveis do *Kernel* actual e de um *bucket*, com a restrição de melhorar o valor da solução obtida anteriormente (caso exista) e de seleccionar pelo menos um hub pertencente ao *bucket*, sendo actualizado o *Kernel*. Este procedimento repete-se sucessivamente enquanto um certo número de *buckets* não tiver sido analisado.

[1] Multi-Period Capacitated Single-Allocation Hub Location Problem with Modular Capacities

Uma vez que o problema em estudo é um problema multi-periódico optou-se por aplicar o esquema heurístico apresentado a cada período de tempo em vez de o aplicar ao problema todo. Deste modo é possível diminuir o tamanho dos problemas MILP a resolver e acelerar o processo de obtenção de uma solução. Como a solução obtida para um período influencia a solução dos períodos seguintes, os períodos de tempo são analisados sequencialmente e a solução obtida para cada período é adicionada como uma restrição nos períodos seguintes.

Para testar este algoritmo foram usadas instâncias de 15 e 25 nodos do conjunto de dados CAB (*Civil Aeronautics Board*) que representam o comportamento dos passageiros de companhias aéreas nos Estados Unidos da América. Foram também considerados 5 períodos de tempo e dois tipos de capacidades e fluxos.

Para avaliar a qualidade da adaptação da heurística *Kernel Search* ao problema em estudo usaram-se as soluções exactas obtidas resolvendo o modelo apresentado por Alumur et al. [6] com um *general solver*. Concluiu-se que a heurística estudada é capaz de obter soluções de boa qualidade num intervalo de tempo razoável, tal como se pretendia. No entanto, ainda é possível melhorar vários aspectos da abordagem heurística de modo a melhorar os tempos computacionais e o valor das soluções obtidas.

Palavras-chave: Localização de hubs, Multi-Período, *Kernel Search*, *Single-Allocation*

ABSTRACT

A hub is a central facility that works as a flow consolidation point and/or serves as a switching, sorting and transshipment point in many-to-many distribution systems. Hubs are mostly used to take advantage of discount factors associated with the flow consolidation.

In this work a heuristic approach was developed in order to obtain (good) solutions for the *Multi-Period Capacitated Single-Allocation Hub Location Problem with Modular Capacities* in a reasonable amount of time.

The *Multi-Period Capacitated Single-Allocation Hub Location Problem with Modular Capacities* is an extension of the classical hub location problem to the situation where the hub network can be progressively built over time. Each demand node must be allocated to exactly one hub (single-allocation) and the planning horizon is divided in several time periods. Since the hub network is not assumed to be complete (the hubs do not have to be directly connected to each other), its design is a part of the decision making process. The objective is to minimize the sum of all the costs involved.

A Kernel Search algorithm was proposed to tackle this problem. The Kernel Search relies in dividing the variables of the problem into smaller subsets (a Kernel and a set of buckets) and solving restricted MILP problems on those sets.

This heuristic scheme is composed of two phases: the initialization phase and the solution phase. In the initialization phase the kernel and the buckets are defined and a initial MILP problem restricted on the Kernel is solved. In the solution phase a sequence of MILP problems restricted on the current Kernel and a bucket is solved and, after solving each MILP problem, the Kernel updated.

Since the problem studied is a multi-period problem, instead of applying the Kernel Search framework to the whole problem, it was applied to each time period separately, adding the solution of each period as a constraint to the following time periods.

Computational tests were performed using 15 and 25 nodes instances from the CAB (Civil Aeronautics Board) data set.

Keywords: Hub Location, Multi-Period, Kernel Search, Single-Allocation

Contents

List of Tables

List of Algorithms

"*Deus quer, o Homem sonha, a Obra nasce*"
(God wills, Man dreams, the Work is born)

Fernando Pessoa

CHAPTER 1

Introduction

Hubs are central facilities that work as flow consolidation points, or, as defined by Alumur and Kara [3], "special facilities that serve as switching, transshipment and sorting points in many-to-many distribution systems". They are mostly used to take advantage of the discount factors associated with the flow consolidation. Usually, it is also more efficient to join the flows originated from close locations instead of directly linking every Origin-Destination (O-D) pair. Postal services, public transportation and logistics distribution are application areas in which the use of hub networks is much relevant.

For example, in public transportation systems interfaces usually act as hubs, connecting different means of transportation as well as allowing a consolidation and redistribution of passengers (the flow, in this case).

A hub location problem (HLP) consists of locating hub facilities and allocating demand nodes to hubs in order to route the traffic between origin-destination pairs (Alumur et al. [3]). In most cases, the goal is to minimize the total cost involved. As Campbell and O'Kelly [17] emphasize, there are some features that distinguish hub location problems from other location problems:

1. The demand is associated with flows between O-D pairs (instead of individual points);
2. The flows are allowed to go through hubs;
3. Hubs are facilities to be located;
4. There is a benefit or a requirement of routing flows via the hubs;
5. The objective function depends on the locations of the hubs and the routing of the flows;
6. Paths between O-D pairs visit at most 2 hubs;
7. Direct O-D flows are not allowed.

Note that in some studies published in the literature, items 6 and 7 are relaxed (see, for instance Contreras [20]). for instance, when the hub network is incomplete, more than two hubs may have to be visited. On the other hand, if the network is complete and the costs or distances

satisfy the triangular inequality it is cheaper to use at most two hubs when routing the flows. In some problems, it is possible to ship flows directly between O-D pairs.

Hubs can perform different functions, depending on the nature of the problem at hand:

- Switching, sorting or connecting, allowing flows to be redirected (the flow comes through one arc and leaves through another);
- Consolidation or breakbulk: allowing flows to be aggregated or disaggregated;
- Processing that can change the nature of transportation;
- Splitting or combining as for packet switching in telecommunications networks;

The main decisions associated with hub location problems are related to the hub locations and the routing of flows. These decisions also include the links between spokes and hubs, the links that connect each pair of hubs and, only when allowed, the direct links between spokes.

There are several types/variants of HLPs.

Depending on the objective, the problem can be classified as a *p-Hub Median Problem*, a *Fixed-Charge Hub Location Problem*, a *p-Hub Center Problem* or a *Hub Covering Problem* (see, for instance, Contreras [20]).

A *p-Hub Median Problem* aims at locating p hubs minimizing the total flow routing cost. In a *Fixed-charge Hub Location Problem* the number of hubs to install is not known *a priori*. instead, a fixed cost for opening a hub is considered and the objective consists of minimizing the total cost for installing the hubs and routing the flows through the network. *p-Hub Center Problems* seek for the minimization of the maximum of some service/cost measure, given that p hubs are to be installed. In a *Hub Covering Problem* the goal is minimize the total cost for installing the hubs and routing the flows, ensuring that all nodes are covered. A node is said to be covered if it is within a specified distance of a hub.

A HLP can be single or multi-product. In a single-product problem there is only one type of flow or the difference in the flows is not relevant for the problem to study. In a multi-product problem there are two or more relevant types of flows that may share the hubs (Correia et al. [25]).

The demand can be deterministic or stochastic. We are facing deterministic demand when we know it in advance and it is not subject to any kind of uncertainty. If this is not the case, it can be either uncertain or stochastic (Alumur et al. [4]). The value of an uncertain parameter lies within a given set and the probabilities associated with each value are either impossible to find or irrelevant. Stochastic demand refers to the case in which the demand can be described by some probability distribution.

In a hub network different allocation strategies can be considered. If the flow originated at a node must be routed through exactly one hub we have a single-allocation pattern. The multiple-allocation case occurs when each node can be allocated to more than one hub. In 2011, Yaman [57] generalized these concepts by introducing the r-allocation hub location problem, where r is the maximum number of hubs to which each node can be allocated. By setting r to 1 or to the number of hubs to be installed the single or multiple-allocation cases can be obtained, respectively.

One important feature of a HLP concerns the hub level network. This can be complete or incomplete. In the former case, all hubs are directly connected to each other, while in the latter, the connections between hubs are a part of the decision making process.

A HLP can be capacitated or uncapacitated. In an uncapacitated problem, there is no capacity associated with the hubs. In a capacitated problem the flow that can be routed via

each bub is limited. One particular type of capacity that has relevance in practice concerns the existence of modular capacities (Correia et al. [24]). In this case, the capacity of a hub is determined by the capacity of the module (or modules when more than one can be considered) installed at the hub.

Fixed costs can be associated with the hubs. Usually, when they are not considered, there is a fixed number of hubs to install (this are called the p-Hub Location Problems). Other fixed costs can be considered in a HLP such as those associated with modules (in case of modular capacities) or hub links (the connections between hubs), to mention a few.

Most of the literature focusing HLPs consider a static setting. This means that a plan must be devised and implemented in a single step. More recently, some work has been developed on multi-period HLPs. In this case, a planning horizon is considered reflecting the time for fully implementing the network. Typically, it is divided into several time periods and the goal is to find a multi-period plan for locating the hubs and routing the flows. The parameters involved in the problem (e.g. demands) can themselves be time-dependent.

The specific problem studied in this dissertation is the *Multi-Period Capacitated Single-Allocation Hub Location Problem with Modular Capacities* recently introduced by Alumur et al. [6]. This is a problem in which we have an incomplete hub level network thus calling for the corresponding network design decisions. Concerning the costs, we have fixed and variable costs for the hubs, operational costs for the hub links, set-up costs for the capacity modules installed at the hubs, and flow routing costs. The problem consists of determining when and where to install the hubs, the allocation of non-hub nodes to hubs in each time period, the capacity modules to be installed in each hub in each period, the hub links operating in each period and how the flow should be routed through the network.

Alumur et al. [6] introduced a mixed-integer linear programming formulation for this problem and performed a set of computational tests. Those tests show that for moderately sized instances, the problem can be tackled using a general solver. However, for large-sized instances, there is a need to develop customized algorithms or heuristics. This is what is accomplished by the current work. On particular, the purpose of this dissertation is to develop a heuristic for obtaining (hopefully good) feasible solutions for this problem.

The remainder of the thesis is organized as follows. In the next chapter a literature review concerning hub location problems is performed. In Chapter 3 the investigated problem is revisited; in Chapters 4 and 5 a new heuristic is introduced and applied to the problem. Finally, in Chapter 6 the performance of the heuristic is discussed and some conclusions are drawn.

CHAPTER 2

Literature Review

When some flows or commodities must be shipped between origin-destination (O-D) pairs, it is often unwise to directly link all pairs, because it can be unimplementable, inefficient and/or highly costly. Hence the need to use hubs, that are central facilities which act as switching, transshipment and sorting points.

Hub location is an important topic within *Location Science* and it has been studied by many interdisciplinary researchers (operations research, transportation, geography, network design, telecommunications, regional science, economics, etc.) (see Campbell and O'Kelly [17], Alumur and Kara [3] and Contreras [20]).

This chapter intends to present a review of some of the work done in Hub Location Problems in the past in order to show the relevance of this dissertation. For that, some models will be explained and some heuristic approaches presented.

The remainder of this Chapter is organized as follows. In Section 2.1 some problems related to the problem to be studied are discussed. In Section 2.2 some solution approaches to Hub Location Problems are referred. Some conclusions will be drawn in Section 2.3.

2.1 Hub Location Problems

The first mathematical formulation for a hub location problem was presented by O'Kelly [41] in 1987, according to Alumur and Kara [3]. That formulation is a quadratic integer programming one and assumes that the hub network is complete (all hubs are directly linked to each other).

In this model, the number of hubs to be opened, say p, is defined *a priori*. The problem is known as the *Uncapacitated Single Allocation p-Hub Median Problem* (USApHMP), which means that each spoke is allocated to exactly one hub and there are no capacity constraints. The objective is to minimize the flow routing costs.

Due to the relevance of this model for understanding many developments in the area of hub location, it is revisited in this thesis. Consider the following notation:

N : set of nodes in the network, with $n = |N|$;

W_{ij} : units of flow to be sent between nodes i and j ($i, j \in N$, $W_{ii} = 0$ by assumption);

C_{ij} : transportation cost of a unit of flow between node i and node j (i, $j \in N$, $C_{ii} = 0$ by assumption);

p : number of hubs to be installed;

α : discount factor applied to inter-hub connections (for economies of scale, consider $\alpha \leq 1$).

The decision variables are defined as follows:

$$X_{ik} = \begin{cases} 1, & \text{if node } i \in N \text{ is allocated to a hub at } k \in N, \\ 0, & \text{otherwise.} \end{cases}$$

For $i \in N$, $X_{ii} = 1$ indicates that node i is a hub.

The model proposed by O'Kelly [41] is the following.

$$\text{minimize} \quad \sum_{i \in N} \sum_{j \in N} W_{ij} \left(\sum_{k \in N} X_{ik} C_{ik} + \sum_{m \in N} X_{jm} C_{jm} + \alpha \sum_{k \in N} \sum_{m \in N} X_{ik} X_{jm} C_{km} \right) \tag{2.1}$$

$$\text{subject to} \quad (n - p + 1) X_{jj} - \sum_{i \in N} X_{ij} \geq 0, \qquad j \in N, \tag{2.2}$$

$$\sum_{j \in N} X_{ij} = 1, \qquad i \in N, \tag{2.3}$$

$$\sum_{j \in N} X_{jj} = p, \tag{2.4}$$

$$X_{ij} \in \{0, 1\}, \qquad i \, , \, j \in N. \tag{2.5}$$

In the above model, the objective function (2.1) represents the total cost for routing flows; inequalities (2.2) guarantee that a node can only be assigned to a hub (since there are p hubs, there are $n - p$ spokes; also, if a hub is installed, it is allocated to itself, so there are $n - p + 1$ nodes that can be assigned to each hub), constraints (2.3) impose that each node is allocated to exactly one hub, equality (2.4) ensures that exactly p hubs are installed and constraints (2.5) define the domain of the decision variables.

Due to the quadratic (non-convex) nature of the objective function, this formulation is very difficult to solve. Campbell [15, 16], Aykin [10] and Skorin-Kapov et al. [50] linearized this formulation using variables with four indexes. Ernst and Krishnamoorthy [27] reduced the number of indexes to three, reducing, the total number of variables of the problem. This new model is denoted *USApHMP-N* and it also assumes that the hub network is complete. In order to introduce it, some additional notation has to be considered:

χ : discount factor applied to node-hub connections (collection);

δ : discount factor applied to hub-node connections (distribution);

O_i : total flow originated at node i ($O_i = \sum_{j \in N} W_{ij} \ \forall i \in N$);

D_i : total flow destined to node i ($D_i = \sum_{j \in N} W_{ji} \ \forall i \in N$);

d_{ij} : distance (usually Euclidean) between nodes i and j;

The decision variables introduced by Ernst and Krishnamoorthy [27] are defined as follows.

$$X_{ik} = \begin{cases} 1, & \text{if node } i \in N \text{ is linked to a hub at } k \in N, \\ 0, & \text{otherwise;} \end{cases}$$

For $i \in N$, $X_{ii} = 1$ indicates that node i is a hub.

Y_{kl}^i : total amount of flow with origin at node i that is routed via hubs k and l (i, k, $l \in N$);

The new model is, then,

$$\text{minimize} \quad \sum_{i\in N}\sum_{k\in N} d_{ik}X_{ik}(\chi O_i + \delta D_i) + \sum_{i\in N}\sum_{k\in N}\sum_{l\in N} \alpha d_{kl} Y_{kl}^i \tag{2.6}$$

$$\text{subject to} \quad \sum_{k\in N} X_{kk} = p, \tag{2.7}$$

$$\sum_{k\in N} X_{ik} = 1, \qquad i \in N, \tag{2.8}$$

$$X_{ik} \leq X_{kk}, \qquad i,k \in N, \tag{2.9}$$

$$\sum_{l\in N} Y_{kl}^i - \sum_{l\in N} Y_{lk}^i = O_i X_{ik} - \sum_{j\in N} W_{ij} X_{jk}, \qquad i,k \in N, \tag{2.10}$$

$$X_{ik} \in \{0,1\}, \qquad i,k \in N, \tag{2.11}$$

$$Y_{kl}^i \geq 0, \qquad i,k,l \in N. \tag{2.12}$$

The objective function (2.6) minimizes the total transportation costs. Those costs are obtained by summing the transportation costs between nodes and hubs (collection), between hubs and nodes (distribution) and between hubs. Constraint (2.7) ensures that exactly p hubs will be opened. Constraints (2.8) guarantee that each node will be assigned to exactly one hub (since variables X_{ik} are integer $\forall i$, $k \in N$) and constraints (2.9) guarantee that nodes can only be assigned to open hubs. Equations (2.10) are the flow divergence equations for flow from origin i at node k. These equations ensure that if node i is allocated to hub k then the flow with origin in i leaving hub k to any other hub equals O_i minus the total amount of flow with origin in i destined to nodes allocated to hub k. They also ensure that if node i is not allocated to hub k then the difference between the incoming flow with origin in i from other hubs and the outgoing flow with origin in i to any other hub equals the flow with origin in i destined to the nodes allocated to hub k. Note that if k is not a hub then all the variables in the equation will be 0. Constraints (2.11) and (2.12) define the domain of the decision variables.

In 1999, the same authors (Ernst and Krishnamoorthy [28]) extended the formulation to the *Capacitated Single Allocation Hub Location Problem* (CSAHLP). They consider fixed costs for opening hubs and relax the constraint associated with the number of hubs to open.

Let Γ_k be the capacity of hub $k \in N$ and F_k the fixed cost for installing a hub at location $k \in N$. The formulation proposed by Ernst and Krishnamoorthy [28] is, then:

$$\text{minimize} \quad \sum_{i\in N}\sum_{k\in N} d_{ik}X_{ik}(\chi O_i + \delta D_i) + \sum_{i\in N}\sum_{k\in N}\sum_{l\in N} \alpha d_{kl} Y_{kl}^i + \sum_{k\in N} F_k X_{kk} \tag{2.13}$$

$$\text{subject to} \quad \sum_{k\in N} X_{ik} = 1, \qquad i \in N, \tag{2.14}$$

$$X_{ik} \leq X_{kk}, \qquad i,k \in N, \tag{2.15}$$

$$\sum_{i\in N} O_i X_{ik} \leq \Gamma_k X_{kk}, \qquad k \in N, \tag{2.16}$$

$$\sum_{l\in N} Y^i_{kl} - \sum_{l\in N} Y^i_{lk} = O_i X_{ik} - \sum_{j\in N} W_{ij} X_{jk}, \qquad i,k \in N, \tag{2.17}$$

$$X_{ik} \in \{0,1\}, \qquad i,k \in N, \tag{2.18}$$

$$Y^i_{kl} \geq 0, \qquad i,k,l \in N. \tag{2.19}$$

The objective function (2.13) minimizes the total transportation costs and the costs for opening hubs, equalities (2.14) and (2.15) ensure that each node is assigned to exactly one hub, constraint (2.16) is the capacity constraint for each hub, constraints (2.17) are the flow divergence constraints and (2.18) and (2.19) define the domain of the decision variables.

This formulation is incomplete, as Correia et al. [23] noticed in 2010. Since the variables Y^i_{kl} are only restricted by (2.17), if $X_{kk} = 0$ then Y^i_{kl} and Y^i_{lk} can both be positive. That happens because if $X_{kk} = 0$ then the equation (2.17) becomes $\sum_{l\in N} Y^i_{kl} = \sum_{l\in N} Y^i_{lk}$, $i,k \in N$. This way, the formulation allows infeasible solutions to be obtained as optimal solutions. Correia et al. [23] considered a new set of constraints that force variables Y^i_{kl} to be equal to zero if node i is not allocated to hub k.

$$\sum_{l\in N} Y^i_{kl} \leq O_i X_{ik}, \quad i,k \in N. \tag{2.20}$$

Since the problem studied in this dissertation considers incomplete hub networks i.e., the hub network is not assumed to be complete and the hub level network design is part of the decision making process, it is relevant to revisit one of the first formulations for incomplete hub networks with fixed costs associated to the hub links. Alumur et al. [5] proposed, in 2009, a formulation for the *Single-Allocation Incomplete Hub Location with fixed costs network design Problem*. The problem studied was uncapacitated.

Consider the following notation:

- W_{ij} : flow from node $i \in N$ to node $j \in N$;
- c_{ij} : transportation cost between nodes $i \in N$ and $j \in N$;
- O_i : units of flow originated at node i ($O_i = \sum_{j\in N} W_{ij} \forall i \in N$);
- D_i : units of flow destined to node i ($D_i = \sum_{j\in N} W_{ji} \forall i \in N$);
- α : hub-to-hub transportation discount factor;
- F_j : fixed cost of opening a hub at node $j \in N$
- G_{ij} : fixed cost of opening a hub link between hubs $i \in N$ and $j \in N$

The decision variables proposed are defined as follows.

$$X_{ij} = \begin{cases} 1, & \text{if node } i \in N \text{ is allocated to a hub at node } j \in N, \\ 0, & \text{otherwise;} \end{cases}$$

$$X_{ii} = \begin{cases} 1, & \text{if node } i \in N \text{ is a hub,} \\ 0, & \text{otherwise;} \end{cases}$$

Z_{ij} = $\begin{cases} 1, & \text{if a hub link is established between hubs } i \in N \text{ and } j \in N, \\ 0, & \text{otherwise;} \end{cases}$

f_{ij}^k : total amount of flow with origin at node k that is routed via hubs i and j, in the direction from i to j (i, j, $k \in N$);

The model is, then:

$$\text{minimize} \quad \sum_{i\in N}\sum_{k\in N} c_{ik}O_iX_{ik} + \sum_{i\in N}\sum_{j\in N}\sum_{k\in N} \alpha c_{ij}f_{ij}^k + \sum_{i\in N}\sum_{k\in N} c_{ki}D_iX_{ik} + \sum_{k\in N} F_kX_{kk} + \sum_{i\in N}\sum_{j\in N:j>i} G_{ij}Z_{ij} \tag{2.21}$$

$$\text{subject to} \quad \sum_{j\in N} X_{ij} = 1, \qquad i \in N, \tag{2.22}$$

$$X_{ij} \leq X_{jj}, \qquad i,j \in N, \tag{2.23}$$

$$Z_{ij} \leq X_{ii}, \qquad i,j \in N : i < j, \tag{2.24}$$

$$Z_{ij} \leq X_{jj}, \qquad i,j \in N : i < j, \tag{2.25}$$

$$\sum_{j\in N:j\neq i} f_{ji}^k + O_kX_{ki} = \sum_{j\in N:j\neq i} f_{ij}^k + \sum_{l\in N} W_{kl}X_{li}, \quad i,k \in N, \tag{2.26}$$

$$f_{ij}^k + f_{ji}^k \leq O_kZ_{ij}, \qquad i,j,k \in N : i < j, \tag{2.27}$$

$$f_{ij}^k \geq 0, \qquad i,j,k \in N : i \neq j, \tag{2.28}$$

$$X_{ij} \in \{0,1\}, \qquad i,j \in N, \tag{2.29}$$

$$Z_{ij} \in \{0,1\}, \qquad i,j \in N : i < j. \tag{2.30}$$

The objective function (2.21) minimizes the total transportation costs and the costs for installing hubs and hub links, constraints (2.22) and (2.23) ensure that each node is allocated to exactly one open hub, inequalities (2.24) and (2.25) guarantee that hub links are only installed between hubs, equations (2.26) are the flow divergence constraints, inequalities (2.27) ensure that the variables f are only positive in the established hub links and constraints (2.28), (2.29) and (2.30) define the domains of the decision variables.

Throughout the years there have been several developments on the formulations of hub location problems (including the one being studied) and many heuristics have been proposed to achieve good feasible solutions and reduce computational times. Some of them are presented in the next section.

2.2 Solution Approaches for Hub Location Problems

In order to obtain solutions for hub location problems, exact or approximate methods can be used. The exact methods include using solvers and finding bounds to improve the computational times of the same solvers. Most of the hub location problems are $\mathcal{NP}$-Hard. Single-allocation HLPs are $\mathcal{NP}$-Hard even if the location of the hubs is known (see Alumur and Kara [3] and Contreras [20]). If the problems are $\mathcal{NP}$-Hard, hardly will large instances be solved to optimality. In that case, we may resort to heuristics. In this Section some of those heuristics will be presented.

Most of the heuristics proposed for Hub Location Problems focus on Uncapacitated cases and only in the recent years Capacitated problems have been focused (see Farahani et al. [29]).

In 1987, O'Kelly [41] proposed two heuristic approaches to find good feasible solutions for the *(Uncapacitated) Single allocation p-hub median problem* with a quadratic formulation (as presented in Section 2.1). Both heuristics consider all the possible locations for the p hubs and, then, for each set of possibilities, allocate the nodes to the hubs. The first heuristic allocates spokes to the nearest hub whereas the second heuristic allocates them to the first or second nearest hub. The results show that the latter performs better.

In 1996, Campbell [16] proposed two heuristic algorithms for the same problem that derive the solution for the single allocation problem from the solution of the multiple allocation problem. These heuristics differ in the allocation of spokes to hubs.

Heuristics based in Tabu Search (TS) have been proposed by many authors for uncapacitated hub location problems, as Klincewicz in 1992 [36], Skorin-Kapov et al. in 1994 [49] and Calik et al. in 2009 [14]. Tabu Search is a Local Search based meta-heuristic. It prohibits returning to neighbour solutions that have already been visited - hence the name *Tabu* - and accepts solutions that are worst than the actual solution if there are no better ones in its neighbourhood, avoiding, that way, being stuck in local optima solutions. In 2007, Chen [19] proposed a hybrid heuristic combining Tabu Search with Simulated Annealing and, in 2009, Silva and Cunha [48] proposed a multi-start Tabu Search that starts with more than one initial solution.

In 1998 Abdinnour-Helm [1] introduced a hybrid heuristic combining Tabu Search and Genetic Algorithms (GATS) to achieve good feasible solutions for the *Uncapacitated Single Allocation p-Hub Location Problem.*

Other Genetic Algorithms (GA) (a meta-heuristic based on the natural selection process using mutation, crossover and selection operators on a population of solutions) have been applied to uncapacitated hub location problems by Abdinnour-Helm and Venkatavamanan in 1998 [2], Topcuoglu et al. in 2005 [55], Cunha and Silva in 2007 [26] and Takano and Arai in 2009 [53]. In 2012 Lin et al. [38] presented a Genetic Algorithm for the *Capacitated Multiple Allocation p-Hub Median Problem.*

Simulated Annealing (SA) (a Local Search based meta-heuristic that mimics the annealing process in metallurgy, that consists in the heating and controlled cooling of a material to reduce its defects; In this case, when the temperature is high the proportion of solutions that worsen the objective value that is accepted is higher than when the temperature is low) was proposed by Aykin in 1995 [10] (combined with a Greedy algorithm) and by Smith et al. in 1996 [51] for the uncapacitated hub location problem and by Ernst and Krishnamoorthy in 1999 [28] for the capacitated one.

Lagrangean Relaxation based approaches (consists in relaxing some constraints of the problem and adding penalties to the objective function when they are not satisfied) have been studied by Pirkul and Shilling in 1998 [42] and by Contreras et al. in 2009 [21] (for both capacitated and uncapacitated cases).

In 1996, Smith et al. [51] introduced a combination between Lagrangean Relaxation and Branch and Bound. In 1995 Aykin [10] proposed a purely Branch and Bound approach (consists in a systematic enumeration of the solutions in such way that they form a rooted tree; using upper and lower bounds some *branches* of that tree can be discarded, reducing the number of solutions that need to be analyzed) and in 1999 Sasaki et al. [47] proposed combining Branch and Bound with a Greedy Heuristic.

In 2008 Randall [44] proposed an Ant Colony Optimization (ACO) approach (a meta-heuristic based on the behavior of ants to find the shortest path to food) for a *Capacitated Single allocation p-Hub Location Problem.* In 2015, Fernandes [30] also proposed an ACO approach to the *Multi-Product Single Allocation Hub Location Problem* and in 2009 Meyer et al. [40] suggested a combination between ACO and Branch and Bound.

Other approaches studied were the Linear Programming (LP) relaxation (when tight enough can lead to integer solutions) by Skorin-KApov et al. in 1996 [50], by Rodríguez-Martín and Salazar-González in 2006 [45] and by Wagner in 2008 [56].

Greedy approaches have been applied to both capacitated (by Bollapragada et al. in 2005 [13]) and uncapacitated hub location problems (by Campbell in 1996 [16]).

A Greedy Randomized Adaptative Search Procedure (GRASP) was able to solve instances with up to 52 nodes in 1992 (see Klincewicz [36]).

Branch and Cut approaches were suggested by Labbé et al. in 2005 [37] and Rodríguez-Martín and Salazar-González in 2008 [46]. Branch and Price was applied to both capacitated (see Contreras et al. [22]) and uncapacitated problems (see Thomadsen [54]).

In 2015, Gelareh et al. [32] proposed a model for a *Multi-Period Uncapacitated Multiple Allocation Hub Location Problem with a Budget Constraint* on the fixed costs of opening, closing and maintaining hubs and hub links. Given the complexity of the problem, not even small-sized instances can be solved to optimality using general MIP solvers, so the authors propose two different solution methods. One of them is a local search based heuristic. The other is a Benders Decomposition. This technique has been often used in facility location problems and is still quite popular and successful as attested in the recent paper (in 2016) by Castro et al. [18]. The idea is to split the variables into two sets, one with the complicating variables (usually integer) and another with the remaining ones. The method relies on relaxing the complicating variables (master problem) and iterates between solving a master problem to optimality and solving a sub-problem of the original one by fixing the variables of the master problem. The sub-problem generates cuts that are added to the master problem. If the original problem is a minimization problem, the master problem provides lower bounds and the sub-problem upper bounds. These bounds are used to prove the optimality of a solution for the original model.

2.3 Conclusions

As shown in the last Section, many heuristics have been proposed to achieve good feasible solutions for Hub Location Problems. Even so, most of them were proposed for uncapacitated problems and only a few for capacitated ones. Since the problem investigated in this dissertation has modular capacities, even the heuristic approaches for capacitated problems are hard to adapt (since we do not know *a priori* the capacity in each hub). For the same reason, it would be very difficult for Local Search based procedures to find feasible neighborhood solutions.

The development of an heuristic approach such as the one we propose for our problem is then motivated by these aspects and by the fact that no one did it before.

CHAPTER 3

Problem Description

In this Chapter the *Multi-Period Capacitated Single Allocation Hub Location Problem with Modular Capacities* is defined (revisiting the work presented by Alumur et al. [6]).

The problem is an extension of the classical hub location problem to the situation in which the hub network can be progressively built and its capacity expanded over time [6].

In this problem, each non-hub node must be allocated to exactly one hub (single-allocation), only one type of flow is considered (single-product) and the planning horizon is divided into multiple time periods (multi-period). The demand is assumed to be deterministic, fixed and known, modular capacities will be used and the hub network is not assumed to be complete, which means that the hubs do not have to be directly connected to each other and that the design of the network is also a part of the decision making process.

Fixed costs will be considered for the installation of hubs, for the addition of modular capacities in the hubs and for the hub links used in each time period. Unitarian costs for operating the flow in the hubs and for sending flow between nodes will be considered. The flow transferred between hubs will take advantage of discount factor for economies of scale. The objective is to minimize the sum of the costs.

Given a set of locations, the flows between all O-D pairs and the costs, it is intended to determine for each time period the locations of the hubs to install, which capacity modules to install in each hub, the allocations of spokes to hubs, which direct links between hubs will be used and the flows circulating in the network.

The remainder of this Chapter is organized as follows. In Section 3.1 some underlying assumptions will be described and the model proposed by Alumur et al. [6] will be presented. In Section 3.2 some comments will be made and conclusions drawn.

3.1 Optimization Model

Some assumptions and notation have to be introduced before presenting an optimization model for the problem. The basic assumptions underlying the multi-period nature of the problem are the following (as presented by Alumur et al. [6]):

1. The planning horizon is finite and divided into time periods.

2. In each time period it is possible to:
   i) open new hubs
   ii) expand the capacity of the existing hubs
3. During the planing horizon it is not possible to:
   i) close an existing hub
   ii) reduce the operating capacity

   Note that the design of a hub network is being planned from scratch, but it can be progressively built over time.
4. Different hub links may operate in different periods (the hub network may change over time even if the set of hubs remains unchanged).
5. A hub must collect its own out-bound flow and distribute its own in-bound flow. As a consequence, all flows between hubs are discounted.
6. Capacity constraints are applied to the volume of flow entering a hub via collection and transfer. Since hub network decisions are involved, the flow between any O-D pair can be routed through more than two hubs. Flow entering a hub may require additional sorting, so the flow entering a hub via transfer is also included in capacity constraints.
7. Capacities are modular.
   i) a set of modules of different sizes is available for each node
   ii) at most one module can be installed/added in each node and time period
8. In each time period a demand node can send flow to any other demand node, so the hub network to be established must be connected.
9. It is possible to allocate a spoke to different hubs in different time periods.
10. The costs involved include:
    i) fixed setup costs for installing the hubs
    ii) fixed costs for operating hub links
    iii) fixed costs for installing modules in the hubs
    iv) variable operational costs for the flow in the hubs
    v) variable costs for routing flows between adjacent nodes
11. All decisions are made in the beginning/end of the time periods

The model presented by Alumur et al. [6] for the problem is based on the model proposed by Ernst and Krishnamoorthy in 1999 [28] and it is also similar to the formulation presented by Alumur et al. in 2009 [5] for incomplete hub networks.

The notation proposed by Alumur et al. [6] that is also adopted in this dissertation, is the following:

N : set of nodes;

T : set of time periods;

w_{ij}^t : flow from node $i \in N$ to node $j \in N$ in period t$\in T$;

O_i^t : units of flow originated at node i in period t ($O_i^t = \sum_{j \in N} w_{ij}^t), \forall i \in N, t \in T$;

D_i^t : units of flow destined to node i in period t ($D_i^t = \sum_{j \in N} w_{ji}^t), \forall i \in N, t \in T$;

Q_k : set of different module types available for a hub located at node $k \in N$;

Γ_k^q : capacity of module type $q \in Q_k$ available for node $k \in N$;

f_k^t : fixed setup cost for locating a hub at node $k \in N$ in period $t \in T$;

g_{kl}^t : fixed setup cost for operating a hub link between hubs $k \in N$ and $l \in N$ in period $t \in T$;

h_k^{qt} : cost for installing a module of type $q \in Q_k$ at hub $k \in N$ in period $t \in T$;

p_k^t : operational cost per unit of flow for hub $k \in N$ in period $t \in T$;

c_{ij}^t : cost of sending a unit of flow from node $i \in N$ to node $j \in N$ in period $t \in T$;

α : economies of scale discount factor for the flow transferred between hubs;

The decision variables proposed are the following:

$$x_{ik}^t = \begin{cases} 1, & \text{if node } i \in N \text{ is allocated to hub } k \in N \text{ in period } t \in T, \\ 0, & \text{otherwise;} \end{cases}$$

$$x_{kk}^t = \begin{cases} 1, & \text{if node } k \in N \text{ is a hub in period } t \in T, \\ 0, & \text{otherwise;} \end{cases}$$

$$z_{kl}^t = \begin{cases} 1, & \text{if a hub link is operated between hubs } k \in N \text{ and } l \in N \text{ in period } t \in T, \\ 0, & \text{otherwise;} \end{cases}$$

$$u_k^{qt} = \begin{cases} 1, & \text{if a module of type } q \in Q_k \text{ is installed at hub } k \in N \text{ in period } t \in T, \\ 0, & \text{otherwise;} \end{cases}$$

y_{kl}^{it} : total amount of flow with origin at node $i \in N$ that is routed via hubs $k \in N$ and $l \in N$ in period $t \in T$;

The model can be described as:

$$\begin{aligned} \text{minimize} \quad & \sum_{t \in T} \sum_{i \in N} \sum_{k \in N} c_{ik}^t O_i^t x_{ik}^t + \sum_{t \in T} \sum_{i \in N} \sum_{k \in N} \sum_{l \in N} \alpha c_{kl}^t y_{kl}^{it} + \sum_{t \in T} \sum_{i \in N} \sum_{k \in N} c_{ki}^t D_i^t x_{ik}^t \\ & + \sum_{t \in T} \sum_{k \in N} p_k^t \left(\sum_{i \in N} O_i^t x_{ik}^t + \sum_{l \in N} \sum_{i \in N} y_{lk}^{it} \right) + \sum_{t \in T} \sum_{k \in N} \sum_{q \in Q_k} h_k^{qt} u_k^{qt} \\ & + \sum_{t \in T} \sum_{k \in N} f_k^t \left(x_{kk}^t - x_{kk}^{t-1} \right) + \sum_{t \in T} \sum_{k \in N} \sum_{l \in N : k < l} g_{kl}^t z_{kl}^t \end{aligned} \tag{3.1}$$

$$\text{subject to} \quad \sum_{k \in N} x_{ik}^t = 1, \qquad \forall i \in N, t \in T, \tag{3.2}$$

$$x_{ik}^t \leq x_{kk}^t, \qquad \forall i, k \in N, t \in T, \tag{3.3}$$

$$x_{kk}^{t-1} \leq x_{kk}^t, \qquad \forall k \in N, t \in T, \tag{3.4}$$

$$\sum_{q \in Q_k} u_k^{qt} \leq x_{kk}^t, \qquad \forall k \in N, t \in T, \tag{3.5}$$

$$\sum_{i \in N} O_i^t x_{ik}^t + \sum_{l \in N} \sum_{i \in N} y_{lk}^{it} \leq \sum_{\tau=1}^{t} \Gamma_k^q u_k^{qt}, \qquad \forall k \in N, t \in T, \tag{3.6}$$

$$\sum_{l \in N, l \neq k} y_{kl}^{it} - \sum_{l \in N, l \neq k} y_{lk}^{it} = O_i^t x_{ik}^t - \sum_{j \in N} w_{ij}^t x_{jk}^t, \qquad \forall i, k \in N, t \in T, \tag{3.7}$$

$$z_{kl}^t \leq x_{kk}^t, \qquad \forall k, l \in N : k < l, t \in T, \tag{3.8}$$

$$z_{kl}^t \leq x_{ll}^t, \qquad \forall k, l \in N : k < l, t \in T, \tag{3.9}$$

$$y_{kl}^{it} + y_{lk}^{it} \leq O_i^t z_{kl}^t, \qquad \forall i, k, l \in N : k < l, t \in T, \tag{3.10}$$

$$x_{ik}^t \in \{0, 1\}, \qquad \forall i, k \in N, t \in T, \tag{3.11}$$

$$x_{kk}^0 = 0, \qquad \forall k \in N, \tag{3.12}$$

$$z_{kl}^t \in \{0, 1\}, \qquad \forall k, l \in N : k < l, t \in T, \tag{3.13}$$

$$u_k^{qt} \in \{0, 1\}, \qquad \forall k \in N, q \in Q_k, t \in T, \tag{3.14}$$

$$y_{kl}^{it} \geq 0, \qquad \forall i, k, l \in N, t \in T. \tag{3.15}$$

The objective function (3.1) represents the total cost, which includes the transportation costs, the operational costs at the hubs and the costs for installing capacity modules, hubs and hub links. Constraints (3.2) and (3.3) ensure that every node is assigned to exactly one hub in every time period. Inequalities (3.4) guarantee that hubs cannot be uninstalled and inequalities (3.5) guarantee that capacity modules can only be installed in open hubs. Constraints (3.6) state that the flow from spokes and from other hubs can't exceed the installed capacity at each hub. Equations (3.7) define the flow divergence constraints. Constraints (3.8) and (3.9) ensure that hub links can only be installed when both ends of the link are hubs. Inequalities (3.10) are consistency constraints between the variables y and z (if there is flow between two hubs, the link connecting them must be installed and the circulating flow originated in a given node $i \in N$ can't be more than itself). Constraints (3.11)-(3.15) define the domain of the decision variables.

Note that variables $y_{kk}^{it} (\forall i, k \in N, t \in T)$, that represent the flow originated from node i that is routed only through hub k in period t, are only bound by the non-negativity constraint (3.15) and by the capacity constraints (3.6). Since their value is minimized in the objective function, their value will tend to be 0. That situation does not lead to a mistake in the costs calculation because the cost of processing the flow from node i in hub k and period t is already accounted by $\sum_{t \in T} \sum_{k \in N} \sum_{i \in N} p_k^t O_i^t x_{ik}^t$. Therefore, there is only the need to redefine their value after the solution is obtained (post-process their value) and when node i is allocated to hub k in period t. The post-processed values of variables y_{kk}^{it} can be achieved by setting:

$$y_{kk}^{it} = O_i^t - \sum_{l \in N : l \neq k} y_{kl}^{it}, \ \textbf{if } x_{ik}^t = 1, \quad \forall i, k \in N, t \in T \tag{3.16}$$

The model presented applies to the case when the hub network is being built from scratch, but can easily be adapted to the case when an existing hub network is being expanded. For that some additional notation should be considered:

N^0 : set of hubs that are operating before the beginning of the planing horizon;

u_k^0 : operating capacity at hub $k \in N$ before the beginning of the planning horizon;

For the model above to accommodate this situation there is the need to set $x_{kk}^0 = 1$ and replace (3.6) with (3.17) and (3.18).

$$\sum_{i \in N} O_i^t x_{ik}^t + \sum_{l \in N} \sum_{i \in N} y_{lk}^{it} \leq \sum_{\tau=1}^{t} \sum_{q \in Q_k} \Gamma_k^q u_k^{qt}, \quad \forall k \in N \setminus N^0, t \in T \tag{3.17}$$

$$\sum_{i \in N} O_i^t x_{ik}^t + \sum_{l \in N} \sum_{i \in N} y_{lk}^{it} \leq u_k^0 + \sum_{\tau=1}^{t} \sum_{q \in Q_k} \Gamma_k^q u_k^{qt}, \quad \forall k \in N^0, t \in T \tag{3.18}$$

The new constraints (3.17) ensure that for the hubs that were closed before the start of the planning horizon constraints (3.6) remain unchanged. Constraints (3.18) add the existing capacity before the start of the planning horizon to the available capacity in each hub during the planning horizon.

Alumur et al. [6] tested the model formulation using the CAB (Civil Aeronautics Board) data set considering 15 and 25 nodes and 5 time periods. IBM ILOG CPLEX 12.4 was used to solve the problems.

The 15 node instances were solved to optimality and an average of approximately 95 minutes was needed to solve them (being the minimum 31 seconds and the maximum almost 12 hours).

For the 25 node instances, a time limit of 24 hours was set and a MIP tolerance was set to 1.5 %, which means that when the solver has proven that the solution found is at most 1.5 % worst than the best bound found, the program stops. In this case, an average of approximately 11 hours was needed to solve the problem and for 6 of the 16 instances tested the 1.5 % gap was not achieved within 24 hours.

These results and the fact that the problem is $\mathcal{NP}$-Hard show the relevance of developing a heuristic approach to this problem.

A Local Search algorithm is a heuristic approach based upon the notion of neighborhood. A neighborhood can be defined as a set of feasible solutions that differ from the current one by some specific aspect. Typically, one or more neighborhoods can be considered for a problem. In our case examples of neighborhoods are the sets of feasible solutions that differ from the current solution by closing a hub, opening a hub, switching a hub and a non-hub, changing the capacity of an open hub, adding a capacity module to a hub, removing a capacity module from a hub, changing an allocation, switching a hub link with another, adding a hub link and closing a hub link.

It is easy to see that most of the changes to perform in the current solution can lead to infeasible solutions. Consider the following cases. Closing a hub implies reallocating the nodes allocated to the hub to close. However, that reallocation may not be possible if the capacity of each remaining hub is smaller than the flow originated from each node to reallocate. Besides, closing a hub in a certain period also implies closing it in all the previous time periods, increasing the likelihood of infeasibility. For the same reason, switching a hub and a non-hub can lead to infeasible solutions if the capacity installed in the hub to open is smaller than the capacity of the closed hub. Changing or removing the capacity of a hub in a certain time period can lead to infeasible solutions for the same reason or to the need of rearranging the allocations.

In Local Search procedures the quality of the initial solution can highly influence the quality of the final solution. Therefore, it is important to build a good feasible solution if a Local Search procedure is to be used. However, due to the single-allocation factor and to the fact that the capacities are modular, it is not a simple task to find a feasible solution for the problem (considering, as well, that the problem has multiple time periods that are not independent).

Because of the reasons stated above, it is better to develop a heuristic approach that is not based on neighborhood search, such as Local Search, Simulated Annealing or Tabu Search. For

that reason, a Kernel Search algorithm, a heuristic algorithm based on the optimization of the problem restricted to smaller sets of variables will be considered.

3.2 Conclusions

The advantage of considering a multi-period model (instead of a static—single period—one) is that it gives a better description of reality and allows to account better for the costs and capacity availability. The fact that modular capacities are considered also makes the problem more realistic because often facility capacities cannot be expanded in a continuous way but in a discrete one (for example, by employing another person, installing another sorting line or adding a new terminal).

Since solutions with a similar value can have a totally different structure, a local search based procedure alone may not work well. Furthermore, a move from one solution to another (e.g., changing allocations, changing a module, closing a hub, replacing an open hub by a closed one, etc.) can easily lead to an infeasible solution, due to the capacity constraints since the capacities are modular and we are considering a single-allocation pattern.

In order to overcome these difficulties, we decided to choose a totally different type of heuristic scheme. In particular, we propose a Kernel Search algorithm for our problem as we detail in the following chapters.

CHAPTER 4

Kernel Search

Kernel Search is "a heuristic framework based on the idea of identifying subsets of variables and in solving a sequence of MILP (Mixed Integer Linear Programming) problems, each problem restricted to one of the identified subsets of variables" (see Guastaroba and Speranza [34]).

In this chapter, the Kernel Search heuristic is presented and the example of an application will be mentioned.

To identify the subsets of variables it is important to sort them (by likeliness of being non-zero in the optimal solution). Some good indicators of that likeliness can be the information provided by the Linear Programming (LP) relaxation (as Guastaroba and Speranza [34] proposed), by a Lagrangean relaxation (as suggested by Avella et al. [9]), or by any other method.

The idea of dividing variables into smaller subsets in order to analyze (some of) them separately has been used by Balas and Zemel [11], in 1980, (selected a core - small subset of variables - and solved a restricted exact problem on it) and by Pisinger [43], in 1999, (proposed changing the size of the core during the algorithm execution) for the Knapsack problem, and by Taillard and Voss [52], in 2001, (proposed a meta-heuristic called POPMUSIC—Partial optimization meta-heuristic under special intensification conditions) for various combinatorial optimization problems.

Kernel Search has been proposed for Portfolio Optimization (see Angelelli et al. [8]), for the multi-dimensional knapsack problem (see Angelelli et al. [7]), for index tracking (see Guastaroba and Speranza [33]) and Bi-objective index tracking (see Filippi et al. [31]), for the Capacitated Facility Location Problem (that will be revisited in this Chapter, see Guastaroba and Speranza [34]) and for Binary Linear Integer Problems, especially applied to the Single Source Capacitated Facility Location Problem (see Guastaroba and Speranza [35]).

From all these problems, the *Capacitated Facility Location Problem* (CFLP) is the most similar to the the one we investigate in this thesis, since it combines integer and non-negative real variables (the real variables depending on the value of the integer ones) and both are location problems. For that reason, the application of Kernel Search to the CFLP will be revisited in this Chapter.

The remainder of this Chapter is organized as follows. In Section 4.1 the Basic Kernel Search heuristic will be presented, in Section 4.2 some variations and enhancements will be introduced and in Section 4.3 the application of Kernel Search to the CFLP will be revisited. Finally, in

Section 4.4 some conclusions will be drawn.

4.1 The Basic Kernel Search Heuristic

As explained before, the Kernel Search (KS) procedure relies on dividing the set of decision variables into smaller sets and then solving a restricted version of the problem on those sets.

Before introducing the algorithm some notation must be discussed. A *promising* variable is a variable that is likely to be positive in the optimal solution of the problem. The *kernel* is the set of promising variables and the *buckets* are the ordered sets of all the remaining variables. The *restricted* MILP problem is a Mixed Integer Linear Programming (MILP) problem restricted to a subset of variables (that means that all the variables outside that subset are set to 0).

In order to distribute the variables between the kernel and the buckets a LP relaxation of the problem is solved. After that, a restricted MILP problem is solved on the initial kernel and then a sequence of restricted MILP problems is solved on the current kernel and a new bucket. After solving each MILP problem the kernel can be updated (either by adding variables, if a variable belonging to a bucket is selected; or removing them, if a variable belonging to the kernel is not selected a certain number of times). This process is outlined in Algorithm 4.1.

Algorithm 4.1 Basic KS: General Scheme

```
1: InitializationPhase
2:     Solve LP relaxation                          ▷ other methods can be used
3:     Build the initial Kernel and sequence of Buckets
4:     Solve a MILP problem on the initial Kernel
5: EndInitializationPhase
6: SolutionPhase
7:     while a certain number of buckets not analyzed do
8:         solve a MILP on the current Kernel and a Bucket
9:         update the current Kernel
10:    end while
11: EndSolutionPhase
```

All feasible solutions obtained by solving a restricted MILP problem are heuristic solutions and provide bounds on the optimal value. Those bounds are used to generate a constraint that will be added to the next restricted MILP to be solved. This procedure ends when all the buckets selected to be analyzed have been analyzed.

4.1.1 Parameters

In this subsection some parameters that will be used in the remainder of this chapter will be introduced.

m : size (number of variables) of the initial kernel;

$lbuck$: size of the buckets;

$NB = \lceil (S - m)/lbuck \rceil$: number of buckets, being S the total number of variables;

$\overline{NB}$: number of buckets that will be analyzed;

rem : number of times a variable can remain in the kernel without being selected;

$Time\_limit$ (optional) : maximum time for the resolution of any restricted MILP problem;

4.2 Variations and Enhancements

In this section the Iterative Kernel Search will be presented (Subsection 4.2.1) and the possibility of fixing variables will be explored (Subsection 4.2.2).

Please note that these variations and enhancements will not be applied to the problem to be studied.

4.2.1 Iterative KS

The main idea of the Iterative Kernel Search is to repeat the Basic Kernel Search Solution Phase while the kernel changes.

The iterative Kernel Search procedure returns solutions whose value is better or equal to the ones provided by the Basic Kernel Search procedure although it may require more computational time.

For more detailed informations about the Iterative Kernel Search algorithm please see Guastaroba and Speranza [34].

4.2.2 Fixing variables

The Basic Kernel Search sets all variables outside the kernel and the current bucket to 0, but there is also the possibility of setting them to 1 (in case they are binary variables).

Binary variables can be fixed to one if their value in the Linear Relaxation of the problem is 1 (it is likely that they will be in the optimal solution) and/or if they are selected a certain number of times by the restricted MILP problems during the Kernel Search procedure (in the same way that variables can be removed using parameter *remove*, a parameter fix can be used to determine when to fix to 1 variables belonging to the kernel).

Fixing variables can help reduce the computational times of each restricted MILP problem since it reduces the solution space of that problem.

4.3 KS for the Capacitated Facility Location Problem

In this Section the work of Guastaroba and Speranza [34] will be revisited.

This Section is organized as follows. In subsection 4.3.1 the formulation of the CFLP will be presented and the model explained, in subsection 4.3.2 the Kernel Search algorithm applied to the CFLP will be revisited and in subsection 4.3.3 the parameters considered in this algorithm will be introduced. Finally, in subsection (4.3.4) some conclusions will be drawn and the result of the parameters optimization will be discussed.

4.3.1 Capacitated Facility Location Problem

In order to better explain the application of Kernel Search to the CFLP we introduce some notation and revisit its well-known optimization model:

I : set of potential locations where a facility can be opened;

J : set of customers;

d_j : demand of customer $j \in J$;

b_i : fixed cost for opening a facility in $i \in I$;

s_i : capacity of facility $i \in I$;

a_{ij} : cost of supplying one unit of demand of customer $j \in J$ from facility $i \in I$;

Without loss of generality, it is assumed that all the parameters above are non-negative (except for the capacities of the facilities that are strictly positive) and that $\sum_{i \in I} s_i \geq \sum_{j \in J} d_j$.

The decision variables should not be confused with the variables of the MP-CSAHLPM. Their names were kept unchanged because they are the usual variables for this classical location problem and can help understand the problem. They are defined as follows:

$$x_i = \begin{cases} 1 & \text{facility } i \in I \text{ is opened,} \\ 0 & \text{otherwise.} \end{cases}$$

y_{ij} = demand of customer $j \in J$ supplied from facility $i \in I$.

The proposed CFLP model is then,

$$\text{minimize} \quad \sum_{i \in I} \sum_{j \in J} a_{ij} y_{ij} + \sum_{i \in I} b_i x_i, \tag{4.1}$$

$$\text{subject to} \quad \sum_{j \in J} y_{ij} \leq s_i x_i, \qquad \forall i \in I, \tag{4.2}$$

$$\sum_{i \in I} y_{ij} = d_j, \qquad \forall j \in J, \tag{4.3}$$

$$y_{ij} \leq d_j x_i, \qquad \forall i \in I, j \in J, \tag{4.4}$$

$$y_{ij} \geq 0, \qquad \forall i \in I, j \in J. \tag{4.5}$$

$$x_i \in \{0, 1\}, \qquad \forall i \in I. \tag{4.6}$$

The objective function (4.1) minimizes the total cost for supplying customers and installing facilities, constraints (4.2) guarantee that the demand supplied by each facility does not exceed its own capacity, equations (4.3) ensure that all the demands are exactly satisfied, inequalities (4.4) are redundant but help provide a tighter LP relaxation and constraints (4.5) and (4.6) define the domain of the decision variables.

4.3.2 The Algorithm

Guastaroba and Esperanza [34] proposed both a Basic Kernel Search procedure and an Iterative Kernel Search procedure for the CPLP. In this subsection only the Basic Kernel Search procedure will be presented.

Since the CPLP includes two sets of variables instead of just one, some adaptations will have to be considered.

It is important to notice that for each $i \in I$ if facility i is closed then it can not supply any demand to any client (i.e. if $x_i = 0$, then $y_{ij} = 0, \forall j \in J$). In this case, we say that the variables y_{ij} are *associated* to the variables x_i.

To build the kernel (and the buckets) the variables x will be considered. Then, for each selected x_i a subset of variables y_{ij} will be considered (also chosen by their values and reduced costs in the LP-Relaxation). Since, some of the y variables will never be considered it is important to check if for all j there is a i such as y_{ij} can be positive (if not, the Kernel Search procedure would not find any feasible solution, because there would be a client whose demand could not be satisfied). This kernel can be denoted by $(K, Y(K))$, and the buckets $B_h := (K_h, Y(K_h)), \forall h = 1, .., NB$.

The detailed algorithm (based on the algorithm presented in Section 4.1) is presented in Algorithm 4.2.

The algorithm is composed of two phases. An initialization phase (that corresponds to lines 3 to 25 in Algorithm 4.2 and lines 1 to 5 in Algorithm 4.1) and a Solution Phase (lines 26 to 37 in Algorithm 4.2 and lines 6 to 11 in Algorithm 4.1). The goal of the Initialization Phase is to build the kernel and the buckets and to solve a first MILP problem restricted on the kernel. The goal of the Solution Phase is to improve the value of the solution (if it exists) solving a restricted MILP on the current kernel and a bucket (a set of variables).

In the Initialization Phase, the linear programming relaxation of the problem is solved (line 2 of Algorithm 4.1 and lines 5 to 12 of Algorithm 4.2). If the LP relaxation if infeasible, then the MILP problem is also infeasible, for it is a particular case of its LP relaxation. For the same reason, if the LP relaxation is integer (in the variables that are supposed to be integer in the MILP problem) then the solution of the LP relaxation is the solution of the problem.

The kernel and the buckets will be built according to what was said previously in this subsection (lines 13 to 19 in Algorithm 4.2 and line 3 in Algorithm 4.1). The facilities are sorted in non-increasing order of the total demand they serve (for those selected) and non-decreasing order of their reduced cost in the LP relaxation for those not selected.

A restricted MILP will be solved on the kernel (lines 20 to 24 in Algorithm 4.2 and line 4 in Algorithm 4.1). If this problem is infeasible, the algorithm is not stopped because, by adding variables to the kernel in the Solution Phase, it is possible that a feasible solution will be found. If the problem is feasible, then we have a feasible solution for the whole problem and its value can be used in the future to bound other restricted MILP problems.

In the Solution Phase a certain number of buckets will be analyzed (lines 27 to 36 in Alg. 4.2 and 7 to 10 in Alg. 4.1). For that, a restricted MILP problem will be solved on the current kernel and a bucket (lines 28 to 34 in Alg. 4.2 and line 8 in Alg. 4.1). In this MILP problem, the value of the best solution found (if any) will be used as a bound and at least one facility from the kernel must be selected (otherwise, the value would not be better than the value of the best solution found). Finally, the kernel must be updated (line 9 in Alg. 4.1 and line 35 in Alg. 4.2).

4.3.3 Parameters

The parameters considered for the Basic Kernel Search procedure in subsection 4.1.1 are the ones considered for its adaptation to the CFLP. The definition of parameters m, $lbuck$ and NB will be slightly different due to the differences in the kernel structure. Another parameter will be considered to define a threshold for selecting the subsets of y variables for each facility.

We have, then:

m : size (number of variables) of set K on the initial kernel;

$lbuck$: size of set $K_h, \forall h = 1..NB-1$ on the buckets;

$NB = \lceil (|I| - m)/lbuck \rceil$: number of buckets, being I the set of potential locations;

γ : threshold for selecting the subset of possible clients for each facility;

Since the selection of the y variables to the subset of possible clients for each facility is based on their likeliness of being non-zero in the optimal solution, this parameter is based on the values of the variables and their reduced costs in the LP relaxation.

In this example, γ was set to be the average of the reduced costs of the y variables.

The variables whose reduced cost does not exceed γ are selected. It is also necessary to check if all clients are assigned to at least one facility.

Algorithm 4.2 Detailed Basic KS for CFLP

Input: a set of facilities and a set of clients
Output: feasible solution (x_H, y_H) or $Fail = TRUE$
InitializationPhase ▷ Build Kernel and Buckets
  $Fail := FALSE$
  Solve LP relaxation and store (x_{LP}, y_{LP}) ▷ other methods can be used
  if LP relaxation Infeasible **then**
    $Fail := TRUE$
    STOP
  end if
  if (x_{LP}, y_{LP}) Integer **then**
    STOP
  end if
  BuildKernelBuckets
    Sort the facilities
    Sort the clients for each facility and select a subset of them
    Build the initial Kernel $(K, Y(K))$: K is composed by the first m sorted facilities and $Y(K)$ by the subsets of customers for each facility
    Build a sequence of Buckets $(B_h, Y(B_h)), h = 1..NB$ with the remaining facilities
    Choose $\overline{NB} \leq NB$
  EndBuildKB
  Solve a MILP problem on the initial Kernel
  if Infeasible **then**
    set $Fail := TRUE$
  else let (x_H, y_H) be the solution and z_H the optimal value of the MILP problem
  end if
EndInitializationPhase
SolutionPhase
  for $h = 1$ to $\overline{NB}$ **do**
    solve a MILP on the current Kernel and a Bucket $((K, Y(K)) \cup (B_h, Y(B_h)))$ with two additional constraints:

  a) z_H as an upper bound for the objective function value;

  b) $\sum_{i \in B_h} x_i \geq 1$;

    if MILP problem feasible **then**
      if $Fail = TRUE$ **then**
        $Fail = FALSE$
      end if
      let (x_H, y_H) be the solution and z_H the optimal value of the MILP problem
    end if
    update the current Kernel by

  a) adding the selected facilities from B_h and the corresponding subset of clients

  b) removing the facilities from K that have not been selected rem times and the correspondent subset of clients

  end for
EndSolutionPhase

4.3.4 Results

Computational tests showed that this procedure is both efficient and effective, since it finds good feasible solutions (and several times the optimal solution, or a better solution than the best know solution, in the cases where no optimal solution is known) in a small amount of CPU time.

The parameter optimization made by Guastaroba and Speranza [34] will be presented and some of the results will be applied in the next chapter.

Parameter Optimization

In order to get a good solution quality - time spent ratio, some parameters had to be optimized. Parameters m, *lbuck* and rem can influence the solution quality by allowing more or less flexibility to the process (the bigger they are, the bigger the number of variables considered at each time) and they can influence the computational times, because the bigger they are, the bigger the computational times (because the MILP problem will be more complex, due to the increase in the number of variables). Parameter $\overline{NB}$ also influences the computational times and solution quality by defining the number of MILP problems that the procedure will solve (the bigger the value of $\overline{NB}$, the bigger the computational time and the better the solution quality).

The computational tests performed by Guastaroba and Speranza [34] showed that, for this problem, the most balanced value for m is the number of non-zero facilities in the LP relaxation (this way, it is also more likely that the restricted MILP problem on the kernel is feasible). The best choice for the parameter *lbuck* is to set it to be the same as m.

Parameter rem was tested with $\overline{NB} = NB$. It was showed that with $rem = 1$ some facilities were removed from the kernel too quickly, leading to worst optimality gaps. The computational times were the fastest though. The gaps were similar for $rem = 2$, $rem = min\{2 + \lfloor (NB - 2)/2 \rfloor, NB\}$ and $rem = NB$ even if the computational times increased with the augment in the value of rem.

The parameter $\overline{NB}$ was tested for $\overline{NB} = 0$ (small), $\overline{NB} = 1$ (medium-small), $\overline{NB} = min\{NB, 3 + \lfloor (NB - 3)/2 \rfloor\}$ (medium-large) and $\overline{NB} = NB$ (large). Setting this parameter to be small reduces the computational times enormously but the solutions found are worst. Medium-large and large values slow down the procedure without great improvements in the solution value.

Guastaroba and Speranza [34] found it pertinent to consider the following values:

m (size of set K) = number of non-closed facilities in the LP relaxation.

lbuck (size of sets K_h) = m

rem (number of times a variable remains in the kernel without being selected) = 2

$\overline{NB} = min\{NB, 2\}$

It was concluded that the Kernel Search procedure was robust in terms of solution quality but not in terms of computational times (since they are highly influenced by the parameters).

4.4 Conclusions

In this Chapter we revisited the so-called Kernel Search scheme emphasizing that it has led to good results when applied to classical facility location problems (Capacitated Facility Location

Problem and Single Source Capacitated Facility Location Problem) and also to other well-known combinatorial optimization problems (Knapsack Problem, Multi-dimensional Knapsack Problem, Index Tracking and Binary Linear Integer Programming Problems).

A Kernel Search procedure emerges as a possibility for obtaining feasible solutions for the problem we are studying in this dissertation since (like the CFLP) it contains several sets of decision variables. In the next chapter we adapt that heuristic scheme to our problem.

CHAPTER 5

Kernel Search Applied to our Problem

In this chapter a heuristic approach based on Kernel Search procedure (see Chapter 4) will be proposed.

As it was already discussed, the complexity of the problem we are studying makes the use of Local Search based procedures potentially inefficient, because solutions with very similar structures can have completely different values (and vice-versa).

Given this handicap concerning Local Search based procedures, the Kernel Search framework generically presented in the previous chapter will be considered. The Kernel Search procedure has the advantage of using restricted versions of the problem ensuring that every solution found is feasible.

The remainder of this chapter is organized as follows. In Section 5.1 the idea of the heuristic will be presented in detail and in Section 5.2 the computational tests performed will be described.

5.1 Heuristic Approach

As stated before, the proposed heuristic approach is based on the Kernel Search procedure.

The remainder of this section is organized as follows. The structure of the procedure will be discussed in Subsection 5.1.1, the Kernel definition in Subsection 5.1.2 and, finally, the algorithm will be presented in Subsection 5.1.3.

5.1.1 Time Division

Since we are working with a multi-period problem, all the variables have a index associated with time. The objective is to find a (good) feasible solution that instructs where and when to install hubs, capacity modules and hub links, which allocations should be considered and how to route the flows.

In order to find a (good) feasible solution for this problem using a Kernel Search based procedure two options can be considered:

1. Apply the Kernel Search procedure to the whole problem (with the necessary adaptations) or

2. Repeat the Kernel Search procedure for each time period separately (with the necessary adaptations as well).

In this work the second option was chosen because it decreases the number of binary variables in each restricted MILP problem and, therefore, decreases their complexity and the computational time required for each of them.

For the solutions of the restricted MILP problems to be feasible for the whole problem, the time periods should be considered in order (first the first time period) and the solution found by the Kernel Search procedure for each time period (before the current one) should be added as a constraint in the restricted MILP problems and LP relaxations.

In the MILP problems the variables until the current time period are integer (except for variables y that are continuous), but the variables from the following time periods can be continuous. This way it is ensured that the flows from the following periods are taken into consideration, even if the solution for those periods is not the object of the current restricted MILP problem.

The process fails if no solution can be found for a specific time period (to have a solution for the whole planning horizon it is needed to have a solution for all the time periods).

Algorithm 5.1 presents the Generic Structure of the Heuristic Approach, as described in the previous paragraphs.

Algorithm 5.1 Heuristic's Structure

```
for each t ∈ T do
    if t > 1 then
        add Solution for τ = 1, ..., t − 1 as a constraint
    end if
    Perform the Basic Kernel Search procedure          ▷ With the necessary changes
    if No solution found then
        STOP
    end if
end for
```

5.1.2 Kernel Structure

Since our multi-period HLP contains more sets of variables than the CFLP presented in the previous chapter it is necessary to redefine the Kernel structure. We start by recalling the decision variables of our problem:

$$x_{ik}^t = \begin{cases} 1, & \text{if node } i \in N \text{ is allocated to hub } k \in N \text{ in period } t \in T, \\ 0, & \text{otherwise.} \end{cases}$$

(Note that $x_{kk}^t = 1$ indicates that a hub is installed at node k in period t.)

$$z_{kl}^t = \begin{cases} 1, & \text{if a hub link is operated between hubs } k \in N \text{ and } l \in N \text{ in period } t \in T, \\ 0, & \text{otherwise.} \end{cases}$$

$$u_k^{qt} = \begin{cases} 1, & \text{if a module of type } q \in Q_k \text{ is installed at hub } k \in N \text{ in period } t \in T, \\ 0, & \text{otherwise.} \end{cases}$$

y_{kl}^{it} = total amount of flow with origin at node $i \in N$ that is routed via hubs $k \in N$ and $l \in N$ in period $t \in T$.

In this case, like in the case of the CFLP (see Subsection 4.3.2) if a hub is not installed in a certain time period ($x^t_{kk} = 0, k \in N, t \in T$) then, no spoke can be allocated to it ($x^t_{ik} = 0, \forall i \in N$), no capacity module can be installed there ($u^t_{kq} = 0, \forall q \in Q_k$), no hub link can start or end there ($z^t_{kl} = 0$ and $z^t_{lk} = 0, \forall l \in N$) and no flow can be routed through that hub ($y^{it}_{kl} = 0$ and $y^{it}_{lk} = 0, \forall l, i \in N$). Therefore, we can say that variables x^t_{ik}, u^t_{kq}, z^t_{kl}, z^t_{lk}, y^{it}_{kl} and y^{it}_{lk} are *associated* to variables x^t_{kk}, $\forall i, k, l \in N, q \in Q_k, t \in T$.

It is important to note that the value of variables z and y depend on the opening of two hubs and not just one and that (assuming that the flows between O-D pairs are positive) the graph must be connected. Therefore, selecting just some of these variables in the Kernel (as it was done with the allocations in Section 4.3) could lead to infeasible problems. Because of that, all the z and y variables will be considered in the restricted MILP problems.

The remaining decision variables will be selected for the Kernel (and Buckets) in a similar way to their selection for the Kernel (and Buckets) in the CFLP case. That means that the Kernel will be defined as $(K, X(K), U(K))$, where K is a set of possible locations for the hubs, $X(K)$ is the set of possible allocations to each hub to open and $U(K)$ is the set of possible capacity modules to install in each possible hub. The buckets are defined in a similar way as $B_h := (K_h, X(K_h), U(K_h)), \forall h = 1, ..., NB$.

As in the case of the CFLP, the number of locations in the Kernel (m) and in each bucket (*lbuck*) will be defined by the number of non-zero locations in the LP relaxation for the first period (except when that number is 1; in that case $m := \lceil |N|/2 \rceil$). In each period, these variables will be ordered according to their value in the LP relaxation (non-increasing order) and (when the LP relaxation value is 0) to their reduced costs (in non-decreasing order). Similarly, the allocation and the capacity modules variables will be selected according to a threshold (one for each type of variables). The variables whose reduced cost does not exceed the threshold will be selected. In this case, the thresholds will be the average of the non-zero reduced costs in the LP relaxation for each period.

The LP relaxation referred in the previous paragraph is the LP relaxation of the original problem in the first time period and the LP relaxation of the problem restricted to the solutions found for the previous time periods (the solutions are added as constraints) in the other periods.

5.1.3 The Algorithm

The parameters used in this adaptation of the Kernel Search procedure will be the same used in the Kernel Search for the CFLP (see 4.3.3) with the addition of a parameter (μ) that defines the threshold for selecting capacity module variables.

The Basic Kernel Search procedure will be adapted to find good feasible solutions for each time period, given a solution for the previous time periods (expect for the first period). It is detailed by Algorithm 5.3. This procedure is similar to the Basic Kernel Search procedures presented in Chapter 4 except for the Kernel and Buckets structure (discussed in the previous subsection) and the addition of the solutions from the previous time periods as a constraint.

As in the case of the CFLP, while building the Kernel or removing hubs from the Kernel, the allocations must be checked (because the graph has to be connected in every feasible solution). Therefore, after building the Kernel or removing a possible hub from it in a time period t, it is checked if $\sum_{k \in N} I(x^t_{ik}) \geq 1, \forall i \in N$, where $I(x) = 1$, **iff** $x \in X(K)$ and $I(x) = 0$, otherwise. If a certain location $i \in N$ does not satisfy the inequality, then variables $x^t_{ik}, \forall k \in N, k \neq i$ are added to $X(K)$. This guarantees that the graph is connected and that all locations can be allocated to some hub.

Since there is no point in installing a hub when no capacity module can be installed on it, when the reduced cost of all the module capacity variables for a certain possible hub in a certain

time period is bigger than the defined threshold (μ) the variable with the smallest reduced cost is added to $U(K)$.

Algorithm 5.2 details the whole heuristic procedure. Note that the first LP relaxation (line 3) is a relaxation of the original problem. If it is impossible (lines 4 to 7), then the original problem is also impossible (since any feasible solution of the original problem is a feasible solution of its Linear Programming relaxation). If the x, u and z variables are integer in the LP relaxation (lines 8 to 12), then they are a solution for the original problem because the only difference between a problem and its LP relaxation is the fact that in the LP relaxation the variables do not have to be integer.

Algorithm 5.2 Detailed Heuristic Approach

1: **Input:** problem data
2: **Output:** (x_H, u_H, z_H, y_H) the heuristic solution and v_H its value or $Fail = TRUE$
3: Solve the LP relaxation of the problem
4: **if** LP relaxation infeasible **then**
5: $Fail := True$
6: STOP
7: **end if**
8: **if** LP relaxation integer **then**
9: Save the solution as (x_H, u_H, z_H, y_H) and its value as v_H
10: $Fail := False$
11: STOP
12: **end if**
13: Determine m, $lbuck$ and NB
14: **for** each $t \in T$ **do**
15: Perform the Basic Kernel Search Procedure ▷ see Algorithm 5.3
16: **if** $Fail = TRUE$ **then**
17: STOP
18: **end if**
19: **end for**

After solving the first LP relaxation, the parameters m, $lbuck$ and NB for the Kernel Search are determined according to the explanation given is Subsection 5.1.2 (line 13). The parameter $\overline{NB}$ is not determined since all the buckets will be analyzed ($\overline{NB} = NB$).

Finally, the adapted Kernel Search Procedure will be performed for each time period (lines 14 to 19 in Algorithm 5.2 and Algorithm 5.3). If for any time period no solution is found (lines 16 to 18 in Alg. 5.2) the procedure will stop, since a solution for the whole problem is a solution for each time period.

The adapted Kernel Search procedure (Alg. 5.3) is divided in two phases. The initialization Phase (lines 3 to 24 in Alg. 5.3 and 1 to 5 in Alg. 4.1) pretends to define the remaining parameters, build the Kernel and Buckets and find an initial solution for the current time period.

For that, a LP relaxation is solved (line 2 in Alg. 4.1 and lines 5 to 14 in Alg. 5.3). This LP relaxation is constrained by the solutions for the previous time periods. Similarly to the first LP relaxation, if it is infeasible, then there is no solution for the current time period and consequently, the procedure fails to find a feasible solution for the whole planning horizon. If the variables x, u and z are integer for the current time period, then a solution for the current time period is found.

According to the reduced costs of the variables x and u corresponding to the current time

Algorithm 5.3 Detailed Basic Kernel Search Procedure for time period t

Input: t time period, previous time periods solutions (if $t > 1$), problem data, B-KS parameters
Output: $(x_H^t, u_H^t, z_H^t, y_H^t)$ the heuristic solution and v_H^t its value for the current time period or $Fail = TRUE$
InitializationPhase
    $Fail := TRUE$
    Solve the LP relaxation adding the solutions for $\tau = 1, ..., t-1$ as a constraint **if** $t > 1$
    if LP relaxation infeasible **then**
        $Fail := TRUE$
        STOP
    end if
    if LP relaxation integer **then**
        Save the solution as (x_H, u_H, z_H, y_H) and its value as v_H ▷ solution for the whole problem
        $Fail := False$
        STOP
    end if
    Define the thresholds
    Build Kernel $(K, X(K), U(K))$ and Buckets $B_h := (K_h, X(K_h), U(K_h)), \forall h = 1, ..., NB$
    Solve a restricted MILP problem on the Kernel adding the solutions for $\tau = 1, ..., t-1$ as a constraint **if** $t > 1$
    if restricted MILP feasible **then**
        Save the solution as $(x_H^t, u_H^t, z_H^t, y_H^t)$ and its value as v_H^t
        if $Fail = TRUE$ **then**
            $Fail := FALSE$
        end if
    end if
EndInitializationPhase
SolutionPhase
    for $h = 1$ to $\overline{NB}$ **do**
        Solve a restricted MILP problem on the current Kernel and a Bucket $((K, X(K), U(K)) \cup B_h)$ adding the following constraints:
        a) the solution for $\tau = 1, ..., t-1$ **if** $t > 1$
        b) v_H^t as an upper bound for the objective function value;
        c) $\sum_{k \in N : x_{kk}^t \in B_h} x_{kk}^t \geq 1$;
        if restricted MILP feasible **then**
            Save the solution as $(x_H^t, u_H^t, z_H^t, y_H^t)$ and its value as v_H^t
            if $Fail = TRUE$ **then**
                $Fail := FALSE$
            end if
        end if
        Update the Kernel
    end for
EndSolutionPhase

period in the LP relaxation, the parameters γ and μ are defined (line 15 in Alg. 5.3). The Kernel and Buckets are built as explained in the previous subsection (line 16).

Finally, a restricted MILP on the Kernel is solved. This restricted MILP is also constrained by the value of the variables in the previous periods of time.

The Solution Phase (lines 25 to 36 in Alg. 5.3 and 6 to 11 in Alg. 4.1) pretends to improve the solution found in the Initialization Phase or to find a new one in the case where no solution was found in the Initialization Phase by analyzing each bucket. For that, a restricted MILP problem is solved on the Kernel and a Bucket with some additional constraints (line 27). Those constraints ensure that the solution for the previous time periods remains unchanged, that at least one hub from the bucket is opened and that the value of the best solution found is a bound for the value of the solution to find.

Finally, the Kernel is updated (line 34) by removing the variables that have not been selected for *rem* times and by adding the variables selected from the current bucket.

As may be recalled, there is a need to post-process the variables y_{kk}^{it} (see Section 3.1) by setting:

$$y_{kk}^{it} = O_i^t - \sum_{l \in N: l \neq k} y_{kl}^{it}, \text{ if } x_{ik}^t = 1, \quad \forall i, k \in N, t \in T \tag{5.1}$$

However, if this post-processing is done after each time period solution is found, in the following periods the value of the variables y_{kk}^{it} can not be constrained in the LP relaxations or in the restricted MILP problems. If it is, the problem will be infeasible because of the capacity constraints (3.6).

5.2 Computational Tests

In order to better evaluate the quality of the heuristic developed, the optimal solutions of the problem will be needed, as well as the amount of time used to find them. The optimal solutions will be obtained by solving the problem formulated in Chapter 3.

The heuristic approach presented in the previous section and the model formulation were tested in a 64-bits Windows 10 computer with a AMD A6-6310 APU with AMD Radeon R4 Graphics 1.8 GHz processor and 8 GB of RAM memory.

The model formulation was tested using the solver IBM ILOG CPLEX 12.6 and the heuristic approach was implemented in c++, using the software Microsoft Visual C++ 2010 Express and the same solver.

The remainder of this section is organized as follows. In Subsection 5.2.1 the instances that will be used to test the heuristic approach will be presented and in Subsection 5.2.2 the results will be presented.

5.2.1 Test Instances

To test the heuristic, the CAB (Civil Aeronautics Board) data set (Beasley [12]) will be used. This data set was introduced in 1987 by O'Kelly [41] and is usually used to test hub location problems. This data set is based on airline passenger interactions in the United States of America.

Table 5.1 presents the parameters used to build the test instances. Instances with 15 and 25 nodes will be used and 5 time periods considered.

The flows in the first period are the flows in the OR Library [12] scaled so that they sum one (as usually done in the literature). The flows in the following periods are calculated in two

Table 5.1: Value of the parameters on the CAB data set.

Description	Parameter	Value
Sets:		
Number of Nodes	$\|N\|$	15, 25
Number of Periods	$\|T\|$	5
Flows:		
Flows in the first period	w_{ij}^1	OR Library [12]
Scenario I (increasing)	w_{ij}^t	Increasing with 5%
Scenario II (random)	w_{ij}^t	$\sim U[0.9w_{ij}^{t-1}, 1.2w_{ij}^{t-1}]$
Capacity modules:		
Capacity set I (loose)	Γ_k^q	0.5, 0.75, 1
Capacity set II (tight)	Γ_k^q	0.4, 0.5, 0.6
Costs: (All costs increase by 2% in each period)		
Fixed setup cost	f_k^1	500
Fixed cost of operating a hub link	g_{kl}^1	5% of the length
Cost of installing a capacity module	h_k^{q1}	100 × Module Capacity $\times 0.9^{q-1}$
Operational cost per unit of flow	p_k^1	1
Cost of sending one unit of flow	c_{ij}^1	OR Library [12]
Economies of scale discount factor	α	0.2, 0.4, 0.6, 0.8

different ways, creating two scenarios. In the first scenario, the flows increase 5 % in each period (the flow of the previous period is multiplied by 1.05). In the second, it is considered that the flow can float between 0.9 and 1.2 of its value in the previous period. Therefore, in order to determine the value of the flows in each time period, the flow of the previous period is multiplied by a random value between 0.9 and 1.2.

Two types of capacity sets will be considered. A loose set, with modules of capacity 0.5, 0.75 and 1 and a tight set with modules of capacity 0.4, 0.5 and 0.6. The tight capacity set will only be considered in the 15 node instances, since the computational times required for it are much higher. The same capacity sets will be considered for every possible hub location.

All the costs considered increase 2 % in each time period, which means that the costs from the previous periods will be multiplied by 1.02 to obtain the costs for the current period. In the first period, costs for sending one unit of flow between two locations can be found in the OR Library [12]. There is an economy of scale in the costs of the capacity modules with a 0.9 factor, which means that the unit capacity cost decreases by 10 % with the increase of the capacity.

Table 5.2 explains the characteristics (capacity types, flow types, discount factor and number of nodes) of each instance. As said before, the 25 node instances will not be tested for the tight capacity type, hence they are not numbered.

5.2.2 Computational Results

In order to evaluate the quality of the Kernel Search based heuristic, the gap between the value of the optimal solution and the heuristic solution will be calculated, using the following expression:

$$GAP(\%) = 100 \times \frac{\text{Value of the heuristic Solution} - \text{Value of the Optimal Solution}}{\text{Value of the Optimal Solution}} \tag{5.2}$$

Two tolerances were defined in CPLEX for both the heuristic and exact approaches. Those tolerances were set to 0.00001. One of the tolerances (EpInt) is the tolerance at which a number is considered integer, for example, 5.00001 is considered 5. The other tolerance (EpGap) is the value of the Gap at which a solution is considered optimal.

Table 5.2: Instance Features

Capacities	**Flows**	α	Number of Nodes 15	25
Set I (loose)	Scenario I (increase)	0.2	1	17
		0.4	2	18
		0.6	3	19
		0.8	4	20
	Scenario II (random)	0.2	5	21
		0.4	6	22
		0.6	7	23
		0.8	8	24
Set II (tight)	Scenario I (increase)	0.2	9	-
		0.4	10	-
		0.6	11	-
		0.8	12	-
	Scenario II (random)	0.2	13	-
		0.4	14	-
		0.6	15	-
		0.8	16	-

Additionally, a time limit was set to 6 hours (21600 seconds) for each MILP problem to solve. After this time, the process (of solving the MILP problem, not the whole heuristic approach) is stopped and the best solution found (if any) is selected.

Since not all the exact problems were optimally solved, the tables with the exact solutions show a column called "Status". The numbers in this column correspond to (see [39]):

1 - Optimal solution is available

3 - Model proved infeasible - does not occur

11 - Aborted due to a time limit

102 - Optimal solution within EpGap found

The CPU time is the sum of the time spent in each core of the processor and, therefore, does not represent the actual time that passed between the start of the programs and their end. Because of that, for the exact solutions two different times will be presented: the CPU time and the Elapsed time. The elapsed time represents the actual time spent. It is important to know the difference between them in order to be able to compare the computational times of the heuristic approach (only Elapsed time is displayed) and the exact model.

AVG represents the average of the computational times and gaps for the instances presented.

Table 5.3 presents the solution values and computational times for the exact model and the heuristic algorithm. The heuristic algorithm was tested for parameter $rem = 1$, $rem = 2$ and $rem = 3$. The solution obtained was the same in all the three cases. Only the computational times differed. Therefore, only one column appears for the total cost and for the gap, but three columns are presented for the elapsed time. Elapsed Time et is the time spent by the procedure when $rem = et$. Column NB indicates the number of buckets built during the procedure. The fact that the solution did not change with the increase of parameter rem might be explained by the fact that the number of buckets is usually ≤ 2.

Note that the increase on the value of parameter rem did not improve the value of the objective function.

Table 5.3: Exact and heuristic solutions for 15 node instances with loose capacities

Inst.	Exact Solution				Heuristic Solution					GAP
	Total cost	CPU time (s)	Elapsed time (s)	Status	Total cost	NB	Elapsed time 1 (s)	Elapsed time 2 (s)	Elapsed time 3 (s)	(%)
1	6838.73	3528	1386	102	6899.51	2	85	97	98	0.89
2	7585.02	5527	2324	102	7654.09	1	150	164	165	0.91
3	7962.85	523	233	1	8098.81	2	96	105	105	1.71
4	8148.96	301	131	102	8148.96	7	69	73	73	0
5	6852.41	4724	1947	102	6957.31	2	80	83	83	1.53
6	7600.99	12390	5376	102	7648.13	1	107	121	120	0.62
7	7982.25	1497	756	1	7985.37	2	81	93	93	0.04
8	8162.62	320	181	102	8166.84	7	70	70	70	0.05
AVG	-	3601	1542	-	-	-	92	101	101	0.72

The computational times increased when the parameter rem increased from 1 to 2, which may imply that the size of the restricted MILP problems also increases (it is known that it does not decrease since the original problems are the same and the number of variables removed is the same or smaller).

The computational times required for $rem = 2$ and $rem = 3$ were practically the same, which may mean that the size of the MILP problems did not change with the difference in this parameter. That is the same as saying that no potential hub was excluded from the Kernel in either case. That is a fact when $NB = 1$ since only two restricted MILP problems are solved in each period, which means that with $rem \geq 2$ no possible hub is excluded from the Kernel. For no potential hub to be excluded from the Kernel when $rem = 2$ and $rem = 3$ in the cases when $NB = 2$ all the hubs in the Kernel have to be selected at least in the first or the second restricted MILP solved. For that to happen when $NB = 7$, it means that each potential hub in the Kernel can only not be selected at most one time after it entered the Kernel and before the last MILP problem is solved, in each time period.

Since there is no improvement in the objective value but there is an augment in the computational times by augmenting the value of parameter rem, in the next set of instances this parameter will only be tested for value 1.

It can be seen that the average gap is very small (0.72 %) and that the maximum gap does not reach 2 %, which means that the heuristic approach provides very good solutions for this type of instances. Besides that, the average time needed by the exact model is around 25 minutes and for the heuristic approach only about one minute and a half is required. It is also very important to notice that the Kernel Search algorithm was able to obtain the optimal solution for instance 4 (the one with 15 nodes, loose capacities, increasing flows and $\alpha = 0.8$). For instances 7 and 8 (with 15 nodes, loose capacities, random flows and $\alpha = 0.6$ and $\alpha = 0.8$, respectively) the solutions found by the heuristic procedure present a very small gap (0.04 % and 0.05%, respectively).

The good results obtained for the "easiest" instances hint that for bigger or more complicated instances, as is the case of the instances with a tight capacity set the Kernel Search scheme will be able to obtain good feasible solutions in a reasonable amount of time.

Table 5.4 presents the solution for the 15 node instances with tight capacities.

The values presented between brackets in the status column represent the value of the gap presented by CPLEX when the procedure was stopped for exceeding the time limit. This gap is not the gap presented before in this subsection. It is calculated as follows:

Table 5.4: Exact and heuristic solutions for 15 node instances with tight capacities

Inst.	**Exact Solution**				**Heuristic Solution**			**GAP**
	Total cost	CPU time (s)	Elapsed time (s)	Status	Total cost	NB	Elapsed time (s)	**(%)**
9	6884.63	20581	9273	102	7038.3	2	115	2.23
10	7723.47	66286	21600	11 (0.26%)	7790.56	2	380	0.87*
11	8292.4	61054	21600	11 (0.59%)	8385.07	2	374	1.12*
12	8577.68	65055	21600	11 (0.24%)	8597.01	2	272	0.23*
13	6880.84	12617	4220	1	7005.36	2	137	1.81
14	7719.8	57860	18502	102	7798.5	2	231	1.02
15	8285.82	66430	21600	11 (0.3%)	8396.93	2	357	1.34*
16	8567.86	46602	16055	102	8633.95	2	183	0.77
AVG	-	49560	16811	-	-	-	256	1.17

$$GAP_{CPLEX} = \frac{\text{value of the best integer solution found} - \text{value of the best bound}}{1e^{-10} + \text{value of the best integer solution found}}$$

Note that much more computational time is needed to find the exact solutions for these instances than the required for the instances with loose capacities. If in the instances with loose capacities an average of 25 minutes was needed to obtain the optimal solution, in the instances with tight capacities an average of 4.6 hours was used. Moreover, in four out of eight instances, the solution was not proved to be optimal within the time limit of 6 hours (although the gaps are smaller than 1 %).

In this case, the heuristic algorithm also performs quite well, being the maximum gap 2.23 % and the average gap 1.17 %. It is possible that the real gaps of the instances whose exact solution was not proved to be optimal are bigger, since the value obtained may be an upper bound for the value of the optimal solution. Those gaps are signaled with "*". Nonetheless, the time spent by the Kernel Search algorithm was only 256 seconds (4.3 minutes).

For all the instances two buckets were built, which means that for each time period three restricted MILPs were solved (one for the Kernel alone and two for the updated Kernel and one bucket).

Table 5.5: Exact and heuristic solutions for 25 node instances with loose capacities

Inst.	**Exact Solution**					**Heuristic Solution**			**GAP**
	Total cost	CPU time (s)	Elapsed time (s)	Status	GAP_{CPLEX} (%)	Total cost	NB	Elapsed time (s)	**(%)**
17	6784.77	42398	21600	11	1.51	6887.56	2	4179	1.52
18	7577.21	39461	21600	11	2.51	7612.1	2	10635	0.46
19	8315.83	46272	21600	11	2.78	8401.46	2	4166	1.03
20	8869.95	52067	21600	11	2.4	8940.48	2	8668	0.8
21	7034.74	40265	21600	11	5.92	6866.49	2	3133	-2.39
22	7500.19	33779	18770	102	1.45	7555.07	2	6153	0.73
23	8260.98	43346	21600	11	2.66	8315.55	2	8118	0.66
24	8813.95	43782	19212	102	1.48	8873.89	2	3717	0.68
AVG	-	42671	20947	-	2.59	-	-	6096	0.44

Table 5.5 presents the exact and heuristic solutions obtained for the 25 node instances with loose capacities.

For this instances, the value of the CPLEX parameter EpGap was set to 1.5 % (only for the exact problem; for the restricted MILP problems, it was kept 0.001 %), in order to try to reduce the computational times needed by the exact model to determine the optimal solution.

With the augment of the size of the instances the average time for the heuristic procedure increased to around 1.7 hours. Since the number of buckets (NB) is 2, as it was in the majority of the instances presented before, the number of MILP problems solved is approximately the same. The size of those problems, however, is bigger, since a larger amount of variables is divided into the same number of buckets.

In this case, a better way for decreasing the computational times would be to increase the number of buckets and, therefore, decrease the number of variables in each restricted MILP problem. This approach, however, may lead to inferior quality solutions and even, if NB is big enough, to infeasible solutions.

Even if the heuristic computational times augmented with the size of the instances, the computational times required by the exact model increased much more (for most of the instances, the 6 hours time limit was not enough to obtain a solution whose distance to the optimal solution was proved to be less than 1.5 %) and the gaps between the solutions found is very small (the average gap is 0.44 %).

It is important to notice that the gap presented for instance 21 is negative. That means that the heuristic approach found a solution with a better value than the best solution obtained by the exact method (that was stopped due to the time limit). Besides finding a better solution for this instance, the heuristic algorithm was also able to do that in a shorter amount of time (around 52 minutes instead of 6 hours).

To better evaluate the quality of the heuristic scheme proposed, one more test was performed. The goal of this test is to see how much time CPLEX needs to find a solution with the same quality of the solution found by the heuristic approach. For that, CPLEX's parameter EpGap was set to be the gap between the heuristic solution and the exact solution. This way, CPLEX will stop when the gap is achieved. The results are displayed in Tables 5.6 and 5.7

Table 5.6: Comparison between the heuristic approach and CPLEX on 15 node instances with loose capacities

Inst.	**Heuristic Solution**			**CPLEX**					El. T. C − El. T. H
	Total cost	Elapsed time (s)	GAP (%)	Total cost	CPU time (s)	Elapsed time (s)	GAP C (%)	GAP R (%)	
1	6899.51	85	0.89	6855.79	2709	899	0.82	0.25	814
2	7654.09	150	0.91	7594.2	2853	1054	0.82	0.12	903
3	8098.81	96	1.71	7973.46	95	32	1.48	0.13	-64
4	8148.96	69	0	8148.96	301	131	0	0	62
5	6957.31	80	1.53	6853.08	647	222	1.44	0.01	143
6	7648.13	107	0.62	7600.99	3759	1557	0.6	0	1449
7	7985.37	81	0.04	7982.25	1320	545	0.02	0	464
8	8166.84	70	0.05	8163.71	266	117	0.05	0.01	48
AVG	-	92	0.72	-	1664	632	0.65	0.08	537

Table 5.6 shows the heuristic solutions obtained and the solutions found by CPLEX with the same quality as the heuristic solutions for the 15 node instances with loose capacity sets.

Column "GAP C" represents GAP_{CPLEX} as described before and "GAP R" represents the real gap between the solution obtained and the optimal solution.

It is possible to see that the gap presented by CPLEX is always bigger than the actual gap between the solution found and the optimal solution. This was expected since CPLEX does not use the optimal value to calculate the gap (in fact, if CPLEX could use the optimal value as a lower bound, the problem would be solved already). In fact, two of those times (instances 6 and 7), CPLEX had already found the optimal solution of the problem when EpGap was reached. However, as can be confirmed in table 5.3 it took CPLEX one more hour to confirm that the solution was optimal in instance 6. In instance 7, the extra amount of time was only 3.5 minutes.

This shows that sometimes CPLEX needs more time to prove that a solution is optimal than to find it.

Column "El. T. C − El. T. H" shows the difference in terms of elapsed time between the heuristic approach and CPLEX with the said stopping criterion. Only in instance 3 CPLEX was faster to find a solution than the heuristic algorithm. That is probably due to the fact that the gap was bigger for that instance. Moreover, the average time saved by using the heuristic approach in this case is 537 seconds (9 minutes), being the average time needed by CPLEX 632 seconds (10.5 minutes).

Table 5.7: Comparison between the heuristic approach and CPLEX on 15 node instances with tight capacities

Inst.	**Heuristic Solution**			**CPLEX**					El. T. C - El. T. H
	Total cost	Elapsed time (s)	GAP (%)	Total cost	CPU time (s)	Elapsed time (s)	GAP C (%)	GAP R (%)	
9	7038.3	115	2.23	6911.23	3042	1134	2.22	0.39	1019
10	7790.56	380	0.87*	-	-	-	-	-	-
11	8385.07	374	1.12*	-	-	-	-	-	-
12	8597.07	272	0.23*	-	-	-	-	-	-
13	7005.36	137	1.81	6880.84	3201	976	1.76	0	838
14	7798.5	231	1.02	7719.8	28136	8858	1.02	0	8626
15	8396.93	357	1.34*	-	-	-	-	-	-
16	8633.95	183	0.77	8567.86	22784	7738	0.77	0	7554
AVG	-	256	1.17	-	14291	4677	1.44	0.1	4509

Table 5.7 shows the solutions found by the heuristic approach and by CPLEX with the same quality as the heuristic solutions for the 15 node instances with tight capacities. The instances whose optimal solution was not found by CPLEX within the 6 hours time limit were not tested since the corresponding gap (signaled with "*") is the gap between the heuristic solution and the best integer solution found by CPLEX. Therefore, this gap is lower or equal to the gap between the heuristic solution and the optimal solution. Because of that, those instances would not provide enough information to evaluate the heuristic approach, since the gaps obtained would not be comparable (since they are not referring to the same solution).

The gap calculated by CPLEX is always higher than the real gap between the solution found and the optimal solution, for the reasons explained before. It is also noticeable that for 3 of the 4 solutions found the solution was optimal, even though that was not proved. For those 3 solutions, CPLEX required an average of 2 extra hours to prove that the solution was optimal (confirm in Table 5.4).

Even if CPLEX found better solutions within the required gap, the heuristic algorithm was faster (by 1.25 hours, in average).

CHAPTER 6

Conclusions

In this last chapter a summary of the work done in this dissertation will be presented, some conclusions will be drawn and some improvements and ideas for future work will be referred to.

The summary and conclusions will be presented in Section 6.1 and in Section 6.2 the topics for future work are referred to.

6.1 Summary and Conclusions

In this work, a heuristic algorithm was proposed to find feasible solutions for the *Multi-Period Capacitated Single-Allocation Hub Location Problem with Modular Capacities*. The heuristic scheme presented is based on Kernel Search, a framework that consists in dividing the variables into smaller sets and solving a restricted MILP problem for each set. Since the problem studied is a Multi-Period problem and its variables are time dependent, the Kernel Search procedure was applied for each time period separately instead of being applied to the whole problem in order to further reduce the size of each restricted MILP problem to solve.

Computational tests were performed using the CAB data set with 15 and 25 nodes. Those computational tests were performed using the heuristic algorithm developed and the model for the problem. The latter tests were run in order to evaluate the quality of the heuristic solutions found.

The time required to obtain the optimal solutions shows that the computational time needed is more influenced by the capacity modules type than by the flow scenario chosen, since it is similar for instances with the same capacity type and different flow scenarios and it is different for instances with the same flow scenario and different capacity types. Moreover, the instances with tight capacities require much more time to be optimally solved. For half of the 15 node instances with tight capacities tested, the optimal solution was not found within a 6 hours time limit.

The heuristic approach was able to find good quality solutions in a small amount of time for the instances with 15 nodes. In fact, for both tight and loose capacity sets the average gap is around 1 %. To achieve those gaps, the heuristic approach only needed an average of 92 seconds (1.5 minutes) for the instances with loose capacity sets and 256 seconds (4.3 minutes) for the instances with tight capacity sets. Even if for the instances with loose capacities the time

required by the model to obtain the optimal solutions was reasonable (around 25 minutes, on average), the heuristic approach still performs faster. For the instances with tight capacity sets, this approach is a great advantage, since it allows finding good feasible solutions in a maximum time of 6.3 minutes when for the exact model 6 hours were not enough to find the optimal solution.

For the 25 node instances with loose capacity sets, 6 hours were not enough for CPLEX to find the optimal solution of most of the instances of the problem. The heuristic approach, however, found feasible solutions within an average of 6096 seconds (1.7 hours) and a maximum of 10635 seconds (3 hours). The quality of the solutions found was also good, with an average gap of 0.44 %. For one of the instances considered, the heuristic procedure was even able to find a better solution than the one obtained by CPLEX (requiring less 5 hours).

The computational times increased significantly with the increase in the number of nodes of the instances. That increase may be caused by the increase in the size of the restricted MILP problems. Even if the restricted MILP problems are smaller than the whole problem, their size is still quite big, for some instances. In order to decrease their size and complexity, decreasing the value of parameters m and $lbuck$, and thus increasing NB is an option. This change in the parameters value can, however, lead to worst solutions.

6.2 Future Work

The results obtained by the heuristic approach developed were good, but some improvements are still possible, specially when dealing with bigger instances.

The relevance of analyzing the buckets is not tested in this work. To determine this relevance, a version of the Kernel Search where the Solution Phase is not applied could be tested. Guastaroba and Speranza [34] tested this for the Capacitated Facility Location Problem and concluded that, for the CFLP's case, it is indeed relevant to analyze some of the buckets.

In order for the Kernel Search procedure to be faster, the time limit for each restricted MILP problem could be reduced. This may lead to worse quality solutions, but, as shown in this work, CPLEX needs more time to prove the optimality of a solution than to find it. Another way to accelerate the process would be to test for smaller values of m and $lbuck$, and, consequently, greater values of NB. This way, each MILP problem to solve would be smaller. The number of MILP problems, however, would increase.

Apart from trying different values for the parameters, the way of calculating m and $lbuck$ can also be tuned. In the present work, we consider $m = lbuck$, being m determined by the number of non-zero variables corresponding to hubs in the first time period in the LP relaxation of the problem. The values of these parameters are the same for all time periods. In order to adapt the Kernel Search procedure for each time period, their values can be calculated in different ways, considering the remaining time periods. For example, one way would be to define m as the average number of non-zero variables corresponding to hubs in each time period. Another option would be to adopt different values for these parameters in each time period.

The possibility of fixing variables to 1 when they are selected a certain number of times (referred in Section 4.2) can also be considered. Apparently, such enhancement can lead to a decrease in computational times (by decreasing the complexity of the restricted MILP problems) without a big risk of worsening the quality of the solutions obtained.

The implementation of an improvement algorithm after the Kernel Search scheme could be a good idea to test if the hubs and capacity modules installed could be installed earlier in time to reduce the total costs. In the cases when this improvement algorithm changes the solution obtained by the Kernel Search procedure proposed, the original problem could be solved once

more by fixing the variables corresponding to the hubs and capacities.

Finally, the values further test can be made on the instances with 25 nodes. The heuristic solutions for the instances with tight capacities can be obtained to confirm if the small difference in the computational times between the instances with 15 nodes and tight and loose capacities is maintained. The values of the solutions obtained heuristically can also be used as an upper bound in the exact model to help finding the optimal solution.

Bibliography

[1] S. Abdinnour-Helm. A hybrid heuristic for the uncapacitated hub location problem. *European Journal of Operational Research*, 106(2):489–499, 1998.

[2] S. Abdinnour-Helm and M. A. Venkataramanan. Solution approaches to hub location problems. *Annals of Operations Research*, 78:31–50, 1998.

[3] S. Alumur and B. Y. Kara. Network hub location problems: The state of the art. *European Journal of Operational Research*, 190(1):1–21, 2008.

[4] S. Alumur, S. Nickel, and F. Saldanha-da-Gama. Hub location under uncertainty. *Transportation Research Part B: Methodological*, 46:529–543, 2012.

[5] S. A. Alumur, B. Y. Kara, and O. E. Karasan. The design of single allocation incomplete hub networks. *Transportation Research Part B: Methodological*, 43(10):936 – 951, 2009.

[6] S. A. Alumur, S. Nickel, F. Saldanha-da-Gama, and Y. Seçerdin. Multi-period hub network design problems with modular capacities. *Annals of Operations Research*, 246(1):289–312, 2016.

[7] E. Angelelli, R. Mansini, and M. G. Speranza. Kernel search: A general heuristic for the multi-dimensional knapsack problem. *Computers & Operations Research*, 37(11):2017–2026, 2010.

[8] E. Angelelli, R. Mansini, and M. G. Speranza. Kernel search: A new heuristic framework for portfolio selection. *Computational Optimization and Applications*, 51(1):345–361, 2012.

[9] P. Avella, M. Boccia, A. Sforza, and I. Vasil'ev. An effective heuristic for large-scale capacitated facility location problems. *Journal of Heuristics*, 15(60):597–615, 2008.

[10] T. Aykin. Networking policies for hub-and-spoke systems with application to the air transportation system. *Transportation Science*, 29(3):201–221, 1995.

[11] E. Balas and E. Zemel. An algorithm for large zero-one knapsack problems. *Operations Research*, 28(5):1130–1154, 1980.

[12] J. Beasley. Or library: Hub location. `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/phubinfo.html`, 1990.

[13] R. Bollapragada, J. Camm, U. S. Rao, and J. Wu. A two-phase greedy algorithm to locate and allocate hubs for fixed-wireless broadband access. *Operations Research Letters*, 33(2):134–142, 2005.

[14] H. Calık, S. A. Alumur, B. Y. Kara, and O. E. Karasan. A tabu-search based heuristic for the hub covering problem over incomplete hub networks. *Computers & Operations Research*, 36(12):3088–3096, 2009.

[15] J. F. Campbell. Integer programming formulations of discrete hub location problems. *European Journal of Operational Research*, 72(2):387–405, 1994.

[16] J. F. Campbell. Hub location and the p-hub median problem. *Operations Research*, 44(6):923–935, 1996.

[17] J. F. Campbell and M. E. O'Kelly. Twenty-five years of hub location research. *Transportation Science*, 46(2):153–169, 2012.

[18] J. Castro, S. Nasini, and F. Saldanha-da Gama. A cutting-plane approach for large-scale capacitated multi-period facility location using a specialized interior-point method. *Mathematical Programming*, pages 1–34, 2016.

[19] J. Chen. A hybrid heuristic for the uncapacitated single allocation hub location problem. *Omega*, 35(2):211–220, 2007.

[20] I. Contreras. Hub location problems. In Stefan Nickel Gilbert Laporte and Francisco Saldanha da Gama, editors, *Location Science*, chapter 12, pages 311–344. Springer, Berlin-Heidelberg, 2015.

[21] I. Contreras, J. A. Díaz, and E. Fernández. Lagrangean relaxation for the capacitated hub location problem with single assignment. *OR Spectrum*, 31(3):483–505, 2009.

[22] I. Contreras, J. A. Díaz, and E. Fernández. Branch and price for large-scale capacitated hub location problems with single assignment. *INFORMS Journal on Computing*, 23(1):41–55, 2011.

[23] I. Correia, S. Nickel, and F. Saldanha-da-Gama. The capacitated single-allocation hub location problem revisited: A note on a classical formulation. *European Journal of Operational Research*, 207(1):92–96, 2010.

[24] I. Correia, S. Nickel, and F. Saldanha-da Gama. Single-assignment hub location problems with multiple capacity levels. *Transportation Research Part B: Methodological*, 44:1047–1066, 2010.

[25] I. Correia, S. Nickel, and F. Saldanha-da-Gama. Multi-product capacitated single-allocation hub location problems: formulations and inequalities. *Networks and Spatial Economics*, 14(1):1–25, 2014.

[26] C. B. Cunha and M. R. Silva. A genetic algorithm for the problem of configuring a hub-and-spoke network for a ltl trucking company in brazil. *European Journal of Operational Research*, 179(3):747–758, 2007.

[27] A. T. Ernst and M. Krishnamoorthy. Efficient algorithms for the uncapacitated single allocation p-hub median problem. *Location Science*, 4(3):139–154, 1996.

[28] A. T. Ernst and M. Krishnamoorthy. Solution algorithms for the capacitated single allocation hub location problem. *Annals of Operations Research*, 86:141–159, 1999.

[29] R. Z. Farahani, M. Hekmatfar, A. B. Arabani, and E. Nikbakhsh. Hub location problems: A review of models, classification, solution techniques, and applications. *Computers & Industrial Engineering*, 64(4):1096–1109, 2013.

[30] E. F. C. Fernandes. A heuristic approach for multi-product capacitated single-allocation hub location problem. *Faculdade de Ciências da Universidade de Lisboa*, `http://hdl.handle.net/10451/20825`, 2015.

[31] C. Filippi, G. Guastaroba, and M. G. Speranza. A heuristic framework for the bi-objective enhanced index tracking problem. *Omega*, 2016.

[32] S. Gelareh, R. N. Monemi, and S. Nickel. Multi-period hub location problems in transportation. *Transportation Research Part E: Logistics and Transportation Review*, 75:67 – 94, 2015.

[33] G. Guastaroba and M. G. Speranza. Kernel search: An application to the index tracking problem. *European Journal of Operational Research*, 217(1):54–68, 2012.

[34] G. Guastaroba and M. G. Speranza. Kernel search for the capacitated facility location problem. *Journal of Heuristics*, 18(6):877–917, 2012.

[35] G. Guastaroba and M. G. Speranza. A heuristic for bilp problems: the single source capacitated facility location problem. *European Journal of Operational Research*, 238(2):438–450, 2014.

[36] J. G. Klincewicz. Avoiding local optima in the p-hub location problem using tabu search and grasp. *Annals of Operations Research*, 40(1):283–302, 1992.

[37] M. Labbé, H. Yaman, and Eric Gourdin. A branch and cut algorithm for hub location problems with single assignment. *Mathematical programming*, 102(2):371–405, 2005.

[38] C. Lin, J. Lin, and Y. Chen. The capacitated p-hub median problem with integral constraints: An application to a chinese air cargo network. *Applied Mathematical Modelling*, 36(6):2777–2787, 2012.

[39] CPLEX Manuals. Solution status codes. `https://www.tu-chemnitz.de/mathematik/discrete/manuals/cplex/doc/refman/html/appendixB.html`.

[40] T. Meyer, A. T. Ernst, and M. Krishnamoorthy. A 2-phase algorithm for solving the single allocation p-hub center problem. *Computers & Operations Research*, 36(12):3143–3151, 2009.

[41] M. E. O'kelly. A quadratic integer program for the location of interacting hub facilities. *European Journal of Operational Research*, 32(3):393–404, 1987.

[42] H. Pirkul and D. A. Schilling. An efficient procedure for designing single allocation hub and spoke systems. *Management Science*, 44(12-part-2):S235–S242, 1998.

[43] D. Pisinger. Core problems in knapsack algorithms. *Operations Research*, 47(4):570–575, 1999.

[44] M. Randall. Solution approaches for the capacitated single allocation hub location problem using ant colony optimization. *Computational Optimization and Applications*, 39(2):239–261, 2008.

[45] I. Rodríguez-Martín and J. Salazar-González. An iterated local search heuristic for a capacitated hub location problem. In F. Almeida, M. J. Blesa Aguilera, C. Blum, and V. Moreno, editors, *Hybrid Metaheuristics*, volume 4030 of *Lecture Notes in Computer Science*, pages 70–81. Springer, Berlin-Heidelberg, 2006.

[46] I. Rodríguez-Martín and J. Salazar-González. Solving a capacitated hub location problem. *European Journal of Operational Research*, 184(2):468–479, 2008.

[47] M. Sasaki, A. Suzuki, and Z. Drezner. On the selection of hub airports for an airline hub-and-spoke system. *Computers & Operations Research*, 26(14):1411–1422, 1999.

[48] M. R. Silva and C. B. Cunha. New simple and efficient heuristics for the uncapacitated single allocation hub location problem. *Computers & Operations Research*, 36(12):3152–3165, 2009.

[49] D. Skorin-Kapov and J. Skorin-Kapov. On tabu search for the location of interacting hub facilities. *European Journal of Operational Research*, 73(3):502–509, 1994.

[50] D. Skorin-Kapov, J. Skorin-Kapov, and M. E. O'Kelly. Tight linear programming relaxations of uncapacitated p-hub median problems. *European Journal of Operational Research*, 94(3):582–593, 1996.

[51] K. Smith, M. Krishnamoorthy, and M. Palaniswami. Neural versus traditional approaches to the location of interacting hub facilities. *Location Science*, 4(3):155–171, 1996.

[52] E. D. Taillard and S. Voss. *POPMUSIC — Partial Optimization Metaheuristic under Special Intensification Conditions*, pages 613–629. Springer US, Boston, MA, 2002.

[53] K. Takano and M. Arai. A genetic algorithm for the hub-and-spoke problem applied to containerized cargo transport. *Journal of Marine Science and Technology*, 14(2):256–274, 2009.

[54] T. Thomadsen and J. Larsen. A hub location problem with fully interconnected backbone and access networks. *Computers & Operations Research*, 34(8):2520–2531, 2007.

[55] H. Topcuoglu, F. Corut, M. Ermis, and G. Yilmaz. Solving the uncapacitated hub location problem using genetic algorithms. *Computers & Operations Research*, 32(4):967–984, 2005.

[56] B. Wagner. A note on "location of hubs in a competitive environment". *European Journal of Operational Research*, 184(1):57–62, 2008.

[57] H. Yaman. Allocation strategies in hub networks. *European Journal of Operational Research*, 211(3):442–451, 2011.